

Efficient Nash Equilibrium Computation for Cybersecurity Games

Michael Lanier David Farmer Yevgeniy Vorobeychik
Washington University in St. Louis

Abstract

Game-theoretic analyses of cyber defence often compute equilibria of games whose payoffs exist only as the output of a simulator. Iterative equilibrium-finding methods grow a set of attacker and defender policies and need the payoff of every attacker–defender pair, so they are bottlenecked by payoff estimation: each payoff costs many simulator runs. We introduce Regret-Weighted Payoff Sampling (RWPS), which spends a fixed simulation budget on the payoffs the equilibrium actually depends on and predicts the rest with a model trained on every payoff measured so far. Standard error bounds for estimated games are driven by the worst-estimated payoff, so they cannot credit an estimator that is inaccurate only where accuracy does not matter. We prove a bound that weights payoff errors by the opponent’s equilibrium strategy, a certificate that can be computed from simulated payoffs alone, and a condition under which errors in the predicted payoffs cannot change either player’s regret. On three synthetic general-sum games, one of them a Colonel Blotto game of military resource allocation, the new bounds are four to six times tighter than the standard one, and RWPS finds less exploitable equilibria than minimum-regret-first search, information-gain search and progressive sampling at the same budget. On two cyber-defence simulators, CyGym and a new game whose hosts are LLM agents exposed to prompt injection, it gives the least exploitable equilibria at the smallest budgets.

1 Introduction

Defending a network is a contest between two adaptive parties. An attacker probes for weaknesses and changes tactics when blocked; a defender allocates limited monitoring and response effort without seeing everything the attacker does. Game theory gives a principled way to plan against such an opponent: model the interaction as a game, compute an equilibrium, and deploy the defender’s equilibrium strategy, which no attacker strategy can exploit by more than the equilibrium’s exploitability. For realistic networks the game cannot be written down in closed form. What a pair of strategies is worth to each side is known only by simulating the network under both of them, which makes these *simulation-based security games*. A family of simulators now supports this work. The Cyber Operations Research Gym [37, 38] underpins the CAGE challenges, Yawning Titan [2], CyberBattleSim [22] and FARLAND [24] offer alternatives at different levels of abstraction, and CyGym [19] models cyber defence directly as a two-player general-sum game.

Games of this size are usually solved by growing each player’s strategy set gradually. Policy-space response oracles (PSRO) [18] start from a few attacker and defender policies and repeat three steps. They estimate the payoff of every attacker–defender pair in the current sets, which gives a small matrix game; they solve that matrix game for a mixed equilibrium; and they train one new policy per player as a best response to the opponent’s current mixture, add it to the sets, and start again. The loop stops when neither new policy improves on the mixture, and MetaDOAR [20] scales the best-response step to large networks.

The three steps cost very different amounts. A best response is a bounded reinforcement learning run. Solving the matrix game takes milliseconds with fictitious play [7] or a standard solver [31]. Estimating one payoff, however, means running the full simulator many times under both policies, and the matrix has $n_D n_A$ entries for pools of n_D defender and n_A attacker policies, so the simulation bill grows quadratically over a run. Simulation throughput is already a known bottleneck: CybORG++ [10] was built largely for speed and reports up to a thousandfold faster parallel execution. Faster simulators usually give up fidelity by abstracting host, process and packet detail [24, 37], and that detail is often the realism the analysis needed. We take the other route and keep the simulator, but call it less often.

RWPS spends a fixed simulation budget on the payoffs that can change the equilibrium and predicts the others with a learned model. The model is trained on every payoff measured earlier in the run, so a simulation paid for in one iteration keeps helping in later ones. A score chooses which payoffs to simulate, favouring those the equilibrium depends on and the model is least sure of. The obvious worry is what this costs in equilibrium quality. The standard analysis of estimated games [40] bounds the damage by twice the largest payoff error over *all* entries. That bound is uninformative here, because RWPS is deliberately inaccurate on entries it judges irrelevant, and those are exactly the entries that set the largest error. Judging the method requires a bound that separates errors the equilibrium can feel from errors it cannot. Our contributions are as follows.

1. **Bounds for estimated games that ignore irrelevant errors.** We sharpen the bound of Tuyls et al. [40] so that a player’s regret depends only on payoff errors weighted by the opponent’s equilibrium strategy, and give a signed version in which only two kinds of error can hurt (Lemma 1). Once every strategy has been simulated against the opponent’s equilibrium strategies, a set of $B^* = n_D s_A + n_A s_D - s_D s_A$ entries, errors in the predicted entries cannot affect regret at all (Corollary 2). We also give a certificate computable from simulated payoffs alone (Corollary 1). On the estimator’s own output the new bounds are four to six times tighter than the standard one.
2. **Regret-Weighted Payoff Sampling.** RWPS predicts unsimulated payoffs with a model trained on every payoff measured so far, chooses what to simulate with a score built from bootstrapped equilibrium strategies, and keeps a small amount of random exploration, which the appendix shows is necessary. Inside PSRO it reaches lower exploitability than minimum-regret-first search, information-gain search [15] and progressive sampling at the same budget. The prediction model accounts for 62% of the gain over random sampling and the score for the other 38%, and with the model held fixed the score beats the sampling rule of Sokota et al. [35] on Colonel Blotto.
3. **The Agentic Network Security Game.** ANSG extends CyGym [19] to networks of LLM agents that an attacker can subvert through indirect prompt injection, with general-sum payoffs and partial observability on both sides. On CyGym and ANSG, RWPS reaches the lowest exploitability at the smallest budgets; on ANSG, without its score it needs four simulated payoffs to reach the level it otherwise reaches with one.

2 Related Work

Several simulators target cyber defence. CybORG [37, 38] drives the CAGE challenges. Yawning Titan [2], CyberBattleSim [22] and FARLAND [24] model networks at different levels of fidelity, and CyGym [19] adds the game-theoretic layer this paper builds on. In all of them, evaluating a strategy pair means running the simulator, and that cost is a recognised obstacle. CybORG++ [10] reports up to a thousandfold parallel speedup from a lightweight re-implementation, built because large-scale

evaluation on the original was infeasible. Our work is complementary to simulator engineering. We reduce the number of simulator calls the equilibrium computation needs, and the two approaches can be combined.

The double-oracle method [21] and PSRO [18] grow strategy pools by best response to the current meta-game equilibrium. Later work improves oracle quality and diversity [5], and studies of the structure of these pools [9] describe what equilibrium supports look like in practice, which matters here because support size sets the cost in Corollary 2. MetaDOAR [20] scales the best-response step to large cyber networks. Little of this work addresses the evaluation step, where most of the cost lies in simulation-based games.

Our theory refines a known result. Tuyls et al. [39, 40] prove that a Nash equilibrium of an estimated meta-game is a 2τ -approximate equilibrium of the true game, where τ is the largest payoff error. Lemma 1 replaces that largest error with one weighted by the opponent’s equilibrium strategy, so the bound stays informative for an estimator that leaves irrelevant cells inaccurate on purpose. Rowland et al. [30] allocate samples for evaluation under incomplete information, mainly for α -rank [27] response graphs, using bandit rules without a surrogate that generalises across cells.

Query-efficient EGTA methods reduce the same cost by pruning profiles instead of predicting them. Progressive-sampling algorithms sample profiles in rounds and stop sampling a profile once its confidence interval shows it cannot affect the learning goal [3, 4, 8, 23]. Our method instead solves a matrix in which every cell is present, either simulated or predicted. The two guarantees promise different things. A uniform guarantee covers the whole estimated matrix, so any equilibrium computed from it is approximately correct, but it exists only after every surviving profile has been sampled to accuracy ε . Our certificate covers only the equilibrium actually returned and says nothing about cells we did not simulate, but it can be computed from whatever has been simulated so far. At the budgets we target, progressive sampling cannot prune a single profile and gives no guarantee (Section 6.1), while our certificate is available at every budget and tightens as more cells are simulated.

Wellman et al. [46] still describe MRFS-style heuristic search as the practical option when exhaustive evaluation is infeasible, so we treat it as a live baseline. We compare against both algorithms of Jordan et al. [15] and reach lower exploitability than both in a growing pool (Section 6.1). Neither carries information between iterations beyond its cache, while our surrogate trains on every cell bought in every earlier iteration, so our advantage grows over a run. On a fixed pool, where there is no such history, MRFS is stronger at low budgets.

On the model-learning side, Wang et al. [44] pair double oracle with a learned game model and regularisation for sample efficiency, although for mean-field games and without an explicit payoff matrix, and Wang et al. [43] study which strategies to add to a pool, a question separate from which profiles to evaluate. Earlier work learns payoff or deviation-payoff functions over strategy spaces [35, 42] within EGTA [41, 45].

Sokota et al. [35] pair a learned deviation-payoff model with sampling around candidate equilibria, which is the idea closest to ours, but they build no payoff table and give no bound. Picheny et al. [28] apply Bayesian optimisation to Nash computation in continuous black-box games with a Gaussian-process surrogate and a fixed game. Adaptive payoff queries for α -rank [29, 30] target a different solution concept. Active learning [33, 34] and deep ensembles [17] provide our acquisition and uncertainty tools. The appendix compares each of these in detail.

3 Method

3.1 Preliminaries and problem statement

A simulation-based security game is a two-player general-sum game between a defender and an attacker. Each player holds a set of pure strategies. In our setting a pure strategy is a complete policy for the underlying stochastic game, for instance a trained neural controller for the defender’s network actions or the attacker’s intrusion actions. The game is *simulation-based* because its payoffs have no analytical form. The expected utility of a strategy pair (d, a) is

$$U^D(d, a) = \mathbb{E}[u(d, a)], \quad U^A(d, a) \text{ symmetrically,}$$

where the expectation is over the simulator’s stochasticity and each sample of it costs a full rollout. The game is general-sum, $U^D + U^A \not\equiv c$, because the two players value the same outcome differently. Nothing below assumes zero-sum structure.

Restricted to finite pools $S_D = \{d_1, \dots, d_{n_D}\}$ and $S_A = \{a_1, \dots, a_{n_A}\}$, the game is a bimatrix game with payoff matrices $D_{ij} = U^D(d_i, a_j)$ and $A_{ji} = U^A(d_i, a_j)$, scaled without loss of generality to $[0, 1]$. A solution is a mixed-strategy Nash equilibrium, a pair of distributions $(p, q) \in \Delta_{n_D} \times \Delta_{n_A}$ from which neither player can profitably deviate,

$$p^\top Dq \geq e_i^\top Dq \quad \forall i, \quad q^\top Ap \geq e_j^\top Ap \quad \forall j.$$

Quality is measured by exploitability, the largest deviation gain either player forgoes,

$$\varepsilon(p, q) = \max \left\{ \max_i (Dq)_i - p^\top Dq, \max_j (Ap)_j - q^\top Ap \right\}, \quad (1)$$

so $\varepsilon = 0$ is an exact equilibrium and, with scaled payoffs, ε reads as a fraction of the payoff range. A profile whose exploitability is at most ε is called an ε -Nash equilibrium.

PSRO solves the full game by repeating three operations. It solves the current restricted game for (p, q) , trains a best response to each opponent mixture with a reinforcement-learning oracle and adds it to the pool, and then extends the payoff matrices with the new strategy’s entries. In the simulation-based setting the first operation is nearly free, the second is a bounded training run, and the third costs N_{MC} rollouts per cell over $\Theta(n_D n_A)$ cells. This paper addresses the third step under a budget. Given pools (S_D, S_A) , a cache of previously simulated entries, and a budget of $B \ll n_D n_A$ new cell evaluations, the goal is to produce estimates $(\hat{D}, \hat{A})$ whose equilibrium is close, in the sense of Eq. (1), to an equilibrium of the true (D, A) .

The design question is which B cells to simulate and what to put in the others.

3.2 The budgeted estimator

Algorithm 1 gives the full procedure. It has four steps, explained below in order: embed each strategy, fit a surrogate, score cells by their influence on the equilibrium, and simulate the top-scoring cells with some forced exploration, which the appendix shows is necessary. We call this *Regret-Weighted Payoff Sampling* (RWPS), because the budget goes to cells in proportion to their estimated influence on equilibrium regret, measured through smoothed bootstrap marginals. The method has two separable parts, the history-trained fill and the acquisition score. Section 6.3 measures each. The fill accounts for most of the gain, and the score matters most at the smallest budgets.

First, each strategy receives an embedding $\varphi(s) \in \mathbb{R}^d$ computed without simulation. RWPS requires only that φ distinguish the strategies in the pool. For policies with an actor μ_π , a natural choice is a behavioural fingerprint over a fixed probe bank $\Xi = \{\xi_1, \dots, \xi_P\}$ of states,

$$\varphi(\pi) = [\text{mean}_p \mu_\pi(\xi_p) \parallel \text{std}_p \mu_\pi(\xi_p)], \quad (2)$$

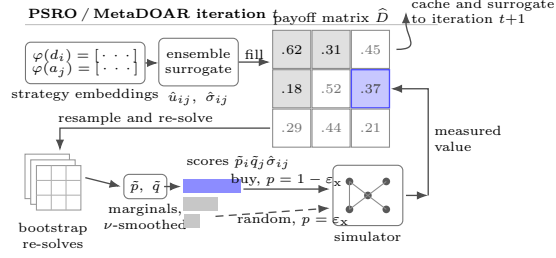


Figure 1: One acquisition round at PSRO / MetaDOAR iteration t . Embeddings feed an ensemble surrogate that fills every uncached entry of $\hat{D}$ (italic) beside the measured ones (grey). Re-solving bootstrap realizations of the filled matrix gives equilibrium marginals, mixed toward uniform at rate ν ; each uncached cell is scored by its marginal weight times the surrogate’s uncertainty, the top-scoring cell is simulated, and its value joins the cache carried into iteration $t+1$.

at the cost of P forward passes. The simplest choice is an identity tag, a one-hot vector per strategy, which carries no behavioural information. Section 5 states the embedding used in each experiment.

Second, an ensemble of M small networks is fit to map embedding pairs to payoff pairs, each on a bootstrap resample of every entry ever simulated, so a payoff bought at one PSRO iteration still trains the surrogate at the next. Queries return the mean $\hat{u}_{ij}$ and disagreement $\hat{\sigma}_{ij}$. Cached entries override surrogate values, and a rank-one row-and-column model stands in until enough labels exist.

Third, each unsimulated cell is scored by how much the equilibrium uses it, times how uncertain the surrogate is about it. We weight by equilibrium use because only those cells can change the answer. By Lemma 1, a player’s regret depends on payoff errors only through the opponent’s equilibrium mixture, and with the support held fixed a change to cell (i, j) moves the defender’s value $p^\top D q$ by $p_i q_j$ times that change. A cell that no equilibrium strategy uses cannot affect the solution however uncertain it is, so scoring by uncertainty alone wastes budget. The weights depend on the unknown equilibrium, but re-solving is nearly free, so we estimate it by re-solving B_{boot} bootstrap realizations of the game, each with a resampled pool and a matrix realization drawn from the surrogate’s predictive distribution.

Weighting by $p_i q_j$ from a single solve would score only cells whose strategies that solve already plays, so an error in the estimated support would never be corrected. We make two choices so that cells outside the estimated support can also be scored. First, we average each player’s mixture across the bootstrap replicates and then multiply, $\bar{p}_i \bar{q}_j$, instead of averaging the product within each replicate. If one replicate plays d_1 against a_1 and another plays d_2 against a_2 , averaging the products scores only those two cells, while multiplying the averages also scores the cross pairs (d_1, a_2) and (d_2, a_1) . Those are the cells that matter when the replicates disagree about the support. Second, each averaged mixture is blended with uniform, $\tilde{p} = (1 - \nu)\bar{p} + \nu/|S_D|$, with a hyperparameter ν , so every strategy keeps some weight even when all replicates agree and are wrong. Setting $\nu = 0$ and averaging the products instead recovers value-sensitivity weighting, reported as an ablation in the appendix.

Finally, cells are simulated one selection at a time. With probability $1 - \epsilon_x$ the highest-scoring unsimulated cell is bought, and with probability ϵ_x a cell chosen uniformly at random is bought instead. The random purchases are needed because the surrogate cannot know where it is wrong on cells it has never simulated. Two strategies can share an embedding while differing arbitrarily in payoff, so any rule that consults only the embedding and the cache can miss the cell that matters, as the appendix proves.

The cache persists across PSRO iterations. Cached values override the surrogate, every fit trains

Algorithm 1 Regret-Weighted Payoff Sampling (RWPS)

Require: pools S_D, S_A ; cache $\mathcal{C}$ of simulated entries; budget B ; batch size c ; exploration probability ε_x ; smoothing ν ; bootstrap size B_{boot} ; ensemble size M

- 1: $\varphi(s) \leftarrow$ embedding of every $s \in S_D \cup S_A$ $\triangleright$ no rollouts
- 2: $B \leftarrow B - |\{(i, j) \in \mathcal{C}\}|$ if B is a coverage target $\triangleright$ spend only on the *uncovered remainder*
- 3: **for** $r = 1, \dots, \lceil B/c \rceil$ **do**
- 4: fit ensemble $\{f_\psi^{(m)}\}_{m=1}^M$ on $\{(\varphi(d_i), \varphi(a_j)) \mapsto \mathcal{C}[i, j]\}$
- 5: $(\hat{u}_{ij}, \hat{\sigma}_{ij}) \leftarrow$ ensemble mean, spread for all $(i, j) \notin \mathcal{C}$
- 6: **for** $b = 1, \dots, B_{\text{boot}}$ **do** $\triangleright$ uncertainty of the *solution*
- 7: subsample $k_D \times k_A$ strategies; draw $\tilde{D}_{ij} \sim \mathcal{N}(\hat{u}_{ij}, \hat{\sigma}_{ij}^2)$ on unsimulated cells, pin cached cells
- 8: $(p^{(b)}, q^{(b)}, v^{(b)}) \leftarrow$ Nash solver on the realization
- 9: **end for**
- 10: $\bar{p}_i \leftarrow \text{mean}_b p_i^{(b)}$; $\bar{q}_j \leftarrow \text{mean}_b q_j^{(b)}$ $\triangleright$ product of marginals
- 11: $\tilde{p} \leftarrow (1 - \nu) \bar{p} + \nu/|S_D|$; $\tilde{q} \leftarrow (1 - \nu) \bar{q} + \nu/|S_A|$ $\triangleright$ smooth toward uniform
- 12: $\alpha(i, j) \leftarrow \tilde{p}_i \tilde{q}_j \hat{\sigma}_{ij}$ $\triangleright$ zero on cached cells
- 13: **for** $t = 1, \dots, c$ **do** $\triangleright$ c cells per refit; the surrogate is refit between batches
- 14: with probability ε_x : simulate a cell drawn uniformly at random, unsimulated first, then revisits averaged by running mean $\triangleright$ necessary; see appendix
- 15: otherwise: simulate the unsimulated cell of largest α
- 16: add to $\mathcal{C}$
- 17: **end for**
- 18: **end for**
- 19: $\hat{D}_{ij}, \hat{A}_{ji} \leftarrow \mathcal{C}[i, j]$ if cached, else ensemble mean
- 20: **return** $(\hat{D}, \hat{A})$

on all cells bought earlier, and a coverage-target budget pays only for the part of the deviation-relevant set not already held. Sokota et al.’s neighbourhood sampler moves mass toward the simplex edges, whereas ν -smoothing moves it toward uniform. The appendix gives further implementation details, such as when the surrogate is refit, and all hyperparameters.

4 Theory: an Instance-Dependent Bound

We now ask what an equilibrium of the estimated game $(\hat{D}, \hat{A})$ is worth in the true game (D, A) . The known answer is the sup-norm bound of Tuyls et al. [39, 40]. Writing $\Delta_D = \hat{D} - D$, $\Delta_A = \hat{A} - A$ and $\tau = \max\{\|\Delta_D\|_\infty, \|\Delta_A\|_\infty\}$, any exact equilibrium of the estimated game is a 2τ -equilibrium of the true one. That bound is too loose for a budgeted estimator. Our estimator leaves irrelevant cells inaccurate on purpose, so τ is set by the cells that do not matter, and the bound is correspondingly weak (Section 6). In this section we let the equilibrium determine which errors count, strengthening the bound step by step from a two-sided worst case to a signed bound that can be computed from the data the estimator already holds. Proofs of the two results below are given here; the remaining proofs, and fuller versions of these, are in the appendix.

4.1 Support-weighted perturbation

Lemma 1 (Support-weighted perturbation). *Let $(\hat{p}, \hat{q})$ be an $\varepsilon_{\text{solve}}$ -approximate Nash equilibrium of $(\hat{D}, \hat{A})$. Write $\Delta_D = \hat{D} - D$ and define the opponent-support-weighted errors $\tau_D(\hat{q}) = \max_i(|\Delta_D| \hat{q})_i$*

and $\tau_A(\hat{p}) = \max_j (|\Delta_A| \hat{p})_j$, with $|\cdot|$ entrywise. Then the defender's regret in the true game satisfies

$$\begin{aligned}
\text{reg}_D - \varepsilon_{\text{solve}} &\leq \underbrace{\max_i (-\Delta_D \hat{q})_i + \hat{p}^\top \Delta_D \hat{q}}_{(a) \text{ signed}} \\
&\leq \underbrace{\tau_D(\hat{q}) + \hat{p}^\top |\Delta_D| \hat{q}}_{(b) \text{ two-term}} \\
&\leq \underbrace{2\tau_D(\hat{q})}_{(c) \text{ support-weighted}} \leq \underbrace{2\tau}_{(d) \text{ sup-norm}},
\end{aligned} \tag{3}$$

where $\tau = \max\{\|\Delta_D\|_\infty, \|\Delta_A\|_\infty\}$, and symmetrically for the attacker. Consequently $(\hat{p}, \hat{q})$ is an ε -Nash equilibrium of (D, A) for ε equal to $\varepsilon_{\text{solve}}$ plus the maximum over players of any of (a)–(d). Equality in the last step holds only when a worst-error cell lies in a played column, respectively row.

Proof. For any pure deviation i of the defender, $e_i^\top D \hat{q} = e_i^\top \hat{D} \hat{q} - e_i^\top \Delta_D \hat{q} \leq \hat{p}^\top \hat{D} \hat{q} + (|\Delta_D| \hat{q})_i = \hat{p}^\top D \hat{q} + \hat{p}^\top \Delta_D \hat{q} + (|\Delta_D| \hat{q})_i$, using optimality of $\hat{p}$ in the estimated game and the triangle inequality. Taking the maximum over i and bounding $\hat{p}^\top \Delta_D \hat{q} \leq \tau_D(\hat{q})$ gives the defender's bound; the attacker's is symmetric, and no zero-sum structure is used. Each row of $|\Delta_D| \hat{q}$ is a $\hat{q}$ -weighted average of entries bounded by τ , so $\tau_D(\hat{q}) \leq \tau$: columns outside $\text{supp}(\hat{q})$ contribute nothing however large their error. The signed form follows by dropping absolute values and keeping only the two harmful directions. Appendix A.1 gives the step-by-step version. $\square$

The chain contains four bounds.¹ Form (d) is the sup-norm bound of Tuyls et al. [39, 40], recovered as the loosest link. Form (c) is its support-weighted refinement, and form (b) keeps both terms instead of doubling the larger. Form (a) drops the absolute values, so that only *underestimated deviations* and *overestimated support* contribute.

We use different parts of the chain at different points below, and the parts are not interchangeable. Theorem 1 and Corollary 2 are derived from (b), since the absolute values are what allow a row to be split over simulated and surrogate cells with each class bounded by a magnitude. Corollary 1 instead rests on the sign structure of (a), which allows an optimistic fill. The solver term $\varepsilon_{\text{solve}}$ is computable without ground truth, since it is the regret of $(\hat{p}, \hat{q})$ in the matrices the estimator already holds. We report it alongside the bounds in Section 6. It matters most for form (a), which has no slack to absorb it.

4.2 Finite-budget guarantee

Under the estimator, entries are of two kinds. Simulated cells carry statistical error, and surrogate cells carry representational error. The support-weighted bound splits accordingly. Assume rollouts return truth plus independent σ -sub-Gaussian noise, each simulated cell holds an average of at least m visits, and surrogate cells have absolute error at most β (from regression error, an uninformative embedding, or both; we do not certify β in advance and instead measure it in Section 6). Which cells are simulated, and how often, is chosen by the acquisition rule from earlier samples, so the design is *adaptive*. A fixed-design union bound would be optimistic under adaptive selection. We therefore state the statistical term with a time-uniform concentration bound [14], valid simultaneously for all cells and all stopping times, at the cost of a log log factor in the visit count. The guarantee below then holds however the acquisition rule selects cells.

¹The bound constrains Δ_D , not its origin, so it also covers irrational players and model misspecification. We do not consider either here. Throughout, Δ_D is sampling and surrogate error.

Theorem 1 (Finite-budget guarantee). *With probability at least $1 - \delta$, simultaneously over all cells and all sample-dependent stopping rules, any $\varepsilon_{\text{solve}}$ -approximate equilibrium $(\hat{p}, \hat{q})$ of $(\hat{D}, \hat{A})$ satisfies*

$$\varepsilon \leq \varepsilon_{\text{solve}} + 2 \max_{\pi \in \{\hat{p}, \hat{q}\}} \left[\zeta(m, \delta) + \beta \cdot w_{\text{sur}}(\pi) \right],$$

where $\zeta(m, \delta) = \sigma \sqrt{\frac{2(\log(4n_D n_A / \delta) + \log \log_2(2m))}{m}}$ is a time-uniform confidence radius valid under adaptive sampling, and $w_{\text{sur}}(\hat{q}) = \max_i \sum_{j: (i,j) \text{ surrogate}} \hat{q}_j$ is the largest opponent-support mass any deviation row places on surrogate cells, and symmetrically for $w_{\text{sur}}(\hat{p})$.

Proof. Split $(|\Delta_D| \hat{q})_i$ over the simulated and surrogate cells of row i . The simulated part is at most the concentration radius times $\sum_j \hat{q}_j \leq 1$, by concentration of m -visit averages and a union bound over all $2n_D n_A$ cells; the surrogate part is at most β times the $\hat{q}$ -mass that row i places on surrogate cells, which is $w_{\text{sur}}(\hat{q})$. Take the maximum over rows, apply Lemma 1, and repeat for the attacker; Appendix A.1 expands each step. $\square$

The theorem tells acquisition what to aim for, which is to drive w_{sur} to zero. Representational error enters in proportion to how much equilibrium mass touches surrogate cells, where the sup-norm bound charges it at full strength. A surrogate cell in a column the opponent never plays costs nothing, while one in a played column costs in proportion to that column's weight. RWPS therefore scores cells by equilibrium sensitivity as well as uncertainty. The weight $\tilde{q}_j$ is zero on the columns that cannot make w_{sur} positive, and largest on those that dominate it. Computing w_{sur} needs only a cache query against the returned mixture, so we can check directly whether it has reached zero, and Appendix A.8 builds a terminating coverage iteration on it. Once it vanishes the bound keeps only its statistical term, which Appendix A.3.2 inverts into the budget a target exploitability requires.

4.3 An optimistic certificate

Only two events produce regret. The estimate can *understate* a deviation row, or it can *overstate* the value of the support. Filling unsimulated cells of a player's own matrix at the maximum payoff therefore understates no deviation, which makes the following bound computable without the true matrices. Its proof, like those of the other results in this section, is in Appendix A.1.

Corollary 1 (Computable optimistic certificate). *Let $(\hat{p}, \hat{q})$ be an $\varepsilon_{\text{solve}}$ -approximate equilibrium of estimated matrices $(\hat{D}, \hat{A})$ whose support block $\text{supp}(\hat{p}) \times \text{supp}(\hat{q})$ is simulated, and let $(D^{\text{opt}}, A^{\text{opt}})$ equal the measured values on simulated cells and the maximum payoff elsewhere. If rollout noise is σ -sub-Gaussian and each simulated cell averages m visits, then with probability at least $1 - \delta$,*

$$\varepsilon \leq \max \left\{ \max_i (D^{\text{opt}} \hat{q})_i - \hat{p}^\top \hat{D} \hat{q}, \max_j (A^{\text{opt}} \hat{p})_j - \hat{q}^\top \hat{A} \hat{p} \right\} + 2\zeta_{m,\delta} + \varepsilon_{\text{solve}},$$

where $\zeta_{m,\delta} = \zeta(m, \delta)$ is the same time-uniform radius as in Theorem 1, valid under adaptive sampling.

Every quantity on the right is available at the end of a build. The deviation terms use $(D^{\text{opt}}, A^{\text{opt}})$, the value terms use the estimated matrices $(\hat{D}, \hat{A})$ the solver was given, and $\varepsilon_{\text{solve}}$ is measured on those, so the true matrices never appear. The surrogate error β of Theorem 1 is the one quantity no data can bound, because it concerns cells never simulated. Covering the deviation-relevant set sets $w_{\text{sur}} = 0$ and removes it.

Corollary 2 (Deviation-relevant coverage). *Let $s_D = |\text{supp}(\hat{p})|$, $s_A = |\text{supp}(\hat{q})|$. If every cell of $\{1:n_D\} \times \text{supp}(\hat{q})$ and of $\text{supp}(\hat{p}) \times \{1:n_A\}$ is simulated, then $w_{\text{sur}}(\hat{p}) = w_{\text{sur}}(\hat{q}) = 0$ and the bound of Theorem 1 reduces to its statistical and solver terms, independent of surrogate quality. The set has*

$$B^* = n_D s_A + n_A s_D - s_D s_A$$

cells, against $n_D n_A$ for a full rebuild.

The principle that a profile’s regret depends only on its unilateral deviations is standard in EGTA [45, 46] and underlies MRFS [15]. Setting $s_D = s_A = 1$ recovers its $2n - 1$ confirmation set. We add the closed-form bimatrix count and the link to Theorem 1. In earlier methods the cells outside the deviation set are never evaluated. Here they are *filled* by a surrogate the solver reads, and the corollary shows that the fill cannot reach either player’s regret, which is why predicting unsimulated payoffs is safe where earlier work pruned them. Because the supports it needs are estimated by re-solving bootstrap samples of the game, without any rollouts, B^* also works as a screen before any simulation. A small B^* says the estimator is worth enabling, and a B^* close to $n_D n_A$ says a full rebuild is the better option. The appendix prices a target certificate from B^* and gives the extended discussion.

Exploration is necessary for the asymptotic guarantee. An embedding is a finite summary, so two policies can share one while differing arbitrarily in payoff, and any deterministic acquisition rule that reads only embeddings and the cache can be led never to simulate the cell that matters, as the appendix proves. The uniform draws of Algorithm 1 are the only mechanism that does not consult the embedding.

5 Experimental Setup

The experiments ask whether the estimator finds better equilibria than other budgeted payoff builds inside growing-pool PSRO (Section 6.1), whether that holds on cyber simulators (Section 6.2), and whether the bounds of Section 4 are tight enough to be worth computing (Section 6.4). On the latent-quality games and Blotto, exploitability is measured against the true, computable Nash equilibrium. On CyGym and ANSG no such equilibrium is available, which is why RWPS is needed there in the first place, so we measure exploitability against a full rebuild of a fixed pool instead.

5.1 Games

All are 21×21 (441 cells), payoffs scaled to $[0, 1]$, rollout noise $\sigma = 0.10$ of the range, $N_{\text{MC}} = 4$ episodes per cell evaluation, so a full rebuild costs 1764 episodes. They differ in what the fingerprint encodes and in equilibrium support size, the two quantities the theory says matter.

Latent quality. Strategies carry latent qualities $q_s \sim \mathcal{N}(0, 1)$ with bilinear general-sum payoffs $D_{ij} = q_i^D q_j^A + 0.3q_i^D$, $A_{ij} = -q_i^D q_j^A + 0.3q_j^A$. Equilibrium supports are $(2, 2)$, so $B^* = 80$ cells, 18%. Two embeddings share these payoffs: an informative one, $[q, \tanh q, q^2, \sin q]$, which makes payoff smooth in φ , and an arbitrary one drawn independently of q , which ablates the embedding (appendix).

Colonel Blotto, asymmetric values. $S = 5$ soldiers over $F = 3$ fields, fields worth different amounts to the two players drawn from $[0.5, 2]$, hence general-sum ($\text{std}(D+A) = 0.21$ across cells). The fingerprint is the normalised allocation; payoff is combinatorial in it. Supports are $(12, 12)$, so $B^* = 360$ cells, or 82%, and the theory predicts before any run that this game is expensive.

CyGym and ANSG. CyGym [19] is a network security game in which an attacker compromises nodes by gaining binary access and a defender responds over a partially observed network. The

Agentic Network Security Game (ANSG) extends it to networks whose nodes are LLM agents. Compromise becomes *goal subversion*. An attacker injects instructions into content an agent reads, by indirect prompt injection or malicious tool triggering, so the agent pursues attacker goals while its legitimate task may still complete. Subverted agents can propagate payloads downstream. The defender audits, reprompts, resets or isolates nodes on noisy signals, and the attacker cannot observe resets and must probe to learn which nodes remain subverted. Valuations are independent per player, so the game is general-sum. ANSG keeps CyGym’s per-node state and PSRO machinery. Full details are in the appendix. In the CyGym and ANSG budget sweeps every strategy is embedded by an identity tag.

5.2 Measurements

Exploitability ε , Eq. (4), of the returned mixture in the true game; lower is better, 0 is exact, and scaling makes it a fraction of the payoff range:

$$\varepsilon = \max\left\{\max_i(Dq)_i - p^\top Dq, \max_j(Ap)_j - q^\top Ap\right\}. \quad (4)$$

Bound tightness: realised ε against 2τ , $2\tau_{\text{supp}}$, and the sharp form of Lemma 1, all computed from the method’s own output. Error decomposition and coverage studies are in the appendix.

5.3 Baselines and protocol

Uniform sampling spends the same budget uniformly at random with a mean fill; *full rebuild* simulates every cell, as standard PSRO does. We additionally compare against minimum-regret-first search and information-gain search [15] and the progressive-sampling family (Section 6.1). The latter shows that a uniform ε -approximation guarantee cannot be bought at these budgets. Restricted games are solved by fictitious play on the synthetic games and on CyGym and ANSG by support enumeration with a Lemke–Howson fallback. The solver’s residual regret enters every bound as $\varepsilon_{\text{solve}}$. The growing-pool comparison uses a 5% per-build budget over twelve iterations with pools growing from three to fourteen per side, sixteen seeds; the fixed-pool diagnostic builds of Section 6.4 sweep $\{5, 10, 20, 40, 70\}\%$ of the matrix over sixteen seeds, measuring ε against the full game by zero-lifting the restricted mixture, with best-response episodes counted separately. All runs are deterministic given recorded seeds. The τ -statistics are maxima over 441 cells and therefore heavy-tailed, so we report medians.

6 Results

All aggregates are mean $\pm$ one standard deviation over sixteen seeds unless a caption states otherwise. Every method in a comparison starts from the same pool.

6.1 Exploitability over PSRO Rounds for Synthetic Games

This is the setting the estimator is built for. Pools grow, the cache persists, and every payoff entry costs a rollout. Table 1 reports twelve PSRO iterations at a 5% per-build budget over sixteen seeds, with exploitability measured against the full 21×21 game by zero-lifting the restricted mixture.

The gap between methods builds up over the run. Figure 2 plots the same runs iteration by iteration. All methods begin together, separate over the first four iterations, and keep their ordering after that.

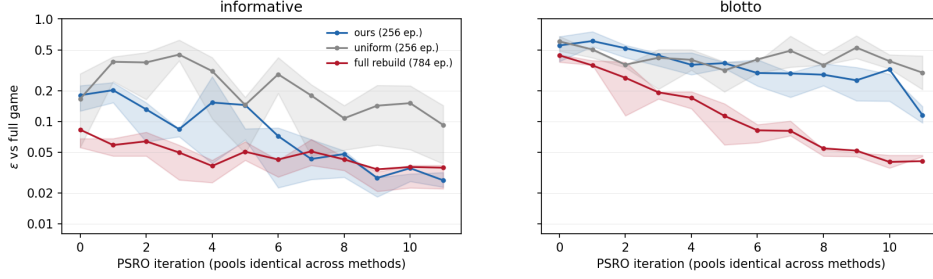


Figure 2: Exploitability against the full game over twelve PSRO iterations at a 5% per-build payoff budget. All methods draw the same initial pool and share a best-response oracle, but each responds to its own mixture, so the pools coincide at the first iteration and diverge thereafter. Means over sixteen seeds, interquartile bands.

By iteration twelve the estimator reaches 0.024 against MRFS’s 0.091 and uniform’s 0.113, and on Blotto 0.082 against 0.222 and 0.293. Both margins are significant under a paired test over the sixteen seeds. Against MRFS, $p = 0.022$ on the informative game and $p < 0.001$ on Blotto, and against uniform, $p = 0.003$ and $p < 0.001$.

RWPS starts behind at iteration 0 (0.262 against 0.180 for MRFS), as expected with an empty cache, and its lead then widens every iteration, which is what reusing earlier simulations should produce.

Against the full rebuild the result divides by game. On the informative game the estimator reaches 0.024 against the rebuild’s 0.036 at a third of the episodes, winning 13 of 16 seeds, but the paired difference is not significant ($p = 0.057$). We claim only that 5% of the cells cost no measurable quality. On Blotto the rebuild is ahead, at 0.045 against 0.082 ($p = 0.004$).

IGS and the progressive-sampling family trail because both need more samples per profile than the budget affords. A 5% budget is ten cell evaluations at a 14×14 pool, against the 294 per cell that PSP’s published schedule requires. The family matches RWPS only at about $100\times$ the full matrix, where PS-REG-M first prunes (Table 2), a budget that defeats the purpose of estimating payoffs when every cell is a simulator rollout.

6.2 Budget sweeps on Cybersecurity Games

On both simulators RWPS reaches the lowest exploitability at the smallest budgets, and the other arms close the gap only as the budget grows (Figures 3 and 4). Uniform sampling and MRFS need most of the available cells to reach the same floor, which RWPS reaches with its first purchase. IGS stays above it throughout, and the progressive-sampling family buys nothing at any budget shown. RWPS leads from the first purchase because every arm starts from the same pre-cached history but only RWPS learns from it, predicting every unbought cell and buying where the equilibrium is sensitive and that prediction is least certain.

The two simulators separate the acquisition score from the fill differently. On CyGym the uniform-acquisition ablation tracks the adaptive rule closely, so the fill carries most of the margin. On ANSG the ablation is well above the floor at budgets 1–3 and reaches it only at budget 4, three purchases after RWPS (adaptive), so the score makes the first few purchases count. This is the cleanest test of the score in the paper, because ANSG keeps every RWPS arm on the rank-one fill and the two variants differ only in which cells they buy. The appendix reports the same sweep on Cyberwheel [26], where the pool is flat and every buying arm ties.

Table 1: Growing-pool PSRO, 5% per-build budget, mean $\pm$ SD over sixteen seeds. ε_0 is the first iteration, where every method holds the same randomly drawn pool. Budget is cumulative payoff-rollout episodes over the run, and every budgeted method is held to the same 256. The full rebuild has no budget. It simulates every uncached cell, costing 784 episodes by iteration twelve, and is shown as a quality reference. The three progressive-sampling members coincide because none prunes a single index at this budget.

Game	Payoff build	ε_0	ε_{12}	budget
Informative	RWPS (ours)	0.262	0.024 ± 0.017	256
	MRFS	0.180	0.091 ± 0.099	256
	IGS	0.168	0.113 ± 0.111	256
	uniform (flat fill)	0.208	0.113 ± 0.097	256
	GS / PSP / PS-REG-M	0.223	0.223 ± 0.174	256
	<i>full rebuild</i>	0.111	0.036 ± 0.023	784
Blotto	RWPS (ours)	0.525	0.082 ± 0.042	256
	MRFS	0.515	0.222 ± 0.114	256
	IGS	0.554	0.276 ± 0.133	256
	uniform (flat fill)	0.593	0.293 ± 0.140	256
	GS / PSP / PS-REG-M	0.627	0.627 ± 0.180	256
	<i>full rebuild</i>	0.398	0.045 ± 0.015	784

Table 2: RWPS against the progressive-sampling family on fixed 21×21 pools, exploitability mean $\pm$ SD over four seeds at budgets that are multiples of the matrix. Bold marks each row’s best. The last column is the share of utility indices PS-REG-M prunes, and the appendix gives all five budgets.

Game	Budget	RWPS	GS	PSP	PS-REG-M	pruned
Inf.	$1 \times$	$.029 \pm .023$	$.162 \pm .138$	$.222 \pm .081$	$.214 \pm .106$	0
	$20 \times$	$.012 \pm .006$	$.413 \pm .125$	$.156 \pm .298$	$.029 \pm .050$	0
	$100 \times$	$.007 \pm .003$	$.025 \pm .040$	$.005 \pm .003$	$.003 \pm .002$	0.5%
Blotto	$1 \times$	$.043 \pm .011$	$.475 \pm .029$	$.471 \pm .026$	$.617 \pm .268$	0
	$20 \times$	$.025 \pm .004$	$.561 \pm .081$	$.366 \pm .081$	$.404 \pm .049$	0
	$100 \times$	$.015 \pm .003$	$.440 \pm .035$	$.013 \pm .001$	$.017 \pm .007$	6.1%

6.3 Acquisition Ablation

Table 1 varies the payoff build as a whole, so its margin mixes two mechanisms, choosing which cells to simulate and predicting the ones we skip. Here we fix the surrogate, budget, batch schedule and exploration rate and vary only the acquisition score, on the same sixteen paired seeds, so any difference between rows is due to acquisition alone.

Uniform here keeps the surrogate fill, while Table 1’s replaces it with a flat value, so comparing the two isolates the fill.

The fill accounts for 62% of the total margin and the acquisition score for 38%. On the informative game the flat-fill baseline reaches 0.113, the surrogate-fill baseline 0.058, and the full method 0.024. On Blotto the same three are 0.293, 0.162, and 0.082. The split holds on both games to the nearest percentage point.

The smoothed rule is first on both games. On Blotto it beats Sokota et al. [35] by 0.038 under a paired t -test ($p = 0.038$) and a signed-rank test ($p = 0.016$), winning 13 of 16 seeds. On the informative game its 0.008 lead is not significant ($p = 0.224$), which we report as a tie.

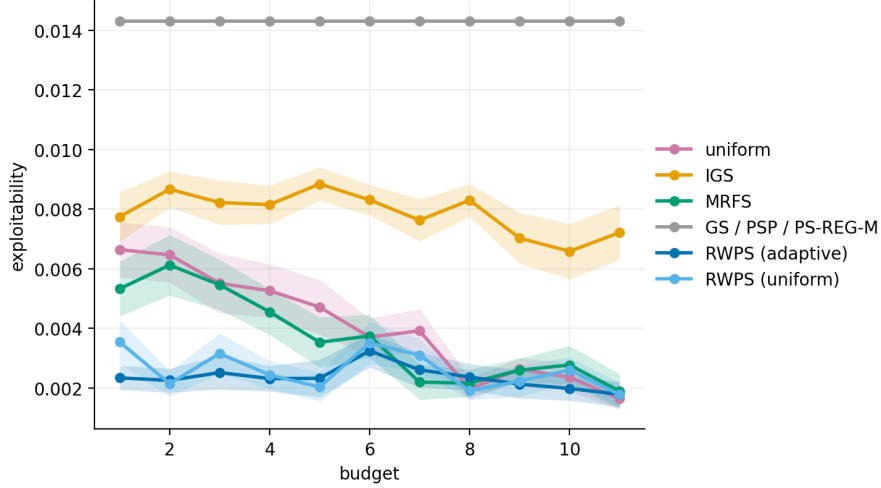


Figure 3: CyGym budget sweep. 6×6 pool with a 5×5 history pre-cached for every arm, leaving 11 cells to buy; 16 seeds. Exploitability is computed against a reference matrix estimated at 16 rollouts per cell. Bands are ± 1 standard error. GS/PSP/PS-REG-M is constant because its sampling schedule requires more samples per cell than the budget affords, so it returns the history-only estimate.

Table 3: Acquisition comparison. All rules share the surrogate, budget, batches and exploration rate and differ only in the score. Mean $\pm$ one standard error of ε_{12} over sixteen paired seeds. Uncertainty sampling scores by surrogate disagreement alone [33, 36], and uniform selection keeps the fill but drops the score. The appendix reports three further variants.

Acquisition score	Informative	Blotto
ν -smoothed product (ours)	0.024 ± 0.004	0.082 ± 0.011
Sokota et al. [35]	0.032 ± 0.004	0.120 ± 0.011
uncertainty ($\hat{\sigma}$ only)	0.040 ± 0.007	0.116 ± 0.013
uniform selection	0.058 ± 0.016	0.162 ± 0.019

Uncertainty sampling beats uniform selection on both games but loses to the full rule ($p = 0.021$ on Blotto, $p = 0.062$ on the informative game). Disagreement shows where the fill is wrong, but a highly uncertain cell can still be irrelevant if no equilibrium strategy plays it.

The appendix sweeps ν , ε_x and the bootstrap sizes.

6.4 Regret Bound Analysis

Errors off the equilibrium support do not affect regret, and the estimator’s own output confirms it. Over all 240 budget-sweep runs every bound holds on every run (Figure 5). Our bound, the signed form (a) of Eq. (3) plus the measured solver term, averages 0.25 of the sup-norm bound, with a median overestimation factor of 2.3 against 11.4. Forms (c) and (b) average 0.47 and 0.33, as the chain predicts. The estimator puts its error on cells no equilibrium strategy plays, those cells set the sup-norm bound, and the support weighting ignores them. The solver term is needed, since without it the signed bound fails on three Blotto runs, each by less than that run’s solver regret. These bounds are diagnostics and cannot be deployed as certificates. Each form of Eq. (3) contains $\hat{D} - D$, so these builds use a fixed-pool bed where the true matrix is available. The appendix gives per-game



Figure 4: ANSG budget sweep. 7×7 pool with a 6×6 history pre-cached for every arm, leaving 13 cells to buy, over 16 seeds. Exploitability is computed exactly against the fully simulated matrix of this pool, so no measurement is clamped. Bands are ± 1 standard error. Two arms are omitted for legibility. GS/PSP/PS-REG-M is constant at 1.97×10^5 payoff units because its sampling schedule requires far more samples per cell than the budget affords, so it buys nothing and returns the history-only estimate. RWPS (uniform), the ablation with the acquisition score removed, is 2.11×10^4 , 1.88×10^4 and 1.57×10^4 at budgets 1–3, and reaches the floor only at budget 4, where RWPS (adaptive) is already at it from budget 1.

factors and the deployable certificate.

7 Conclusion

In simulation-based security games the expensive step is measuring payoffs, and most of those measurements can be avoided. On a 5% per-build budget RWPS found less exploitable equilibria than every other budgeted method we tested, and did best at the smallest budgets on CyGym and ANSG. Our bounds are four to six times tighter than the standard one and need no access to the true payoffs, so an analyst can state how exploitable a computed defence is without ever simulating the full game. Together these make equilibrium analysis affordable at the simulator fidelity the analysis calls for, including on new threat models such as networks of LLM agents.

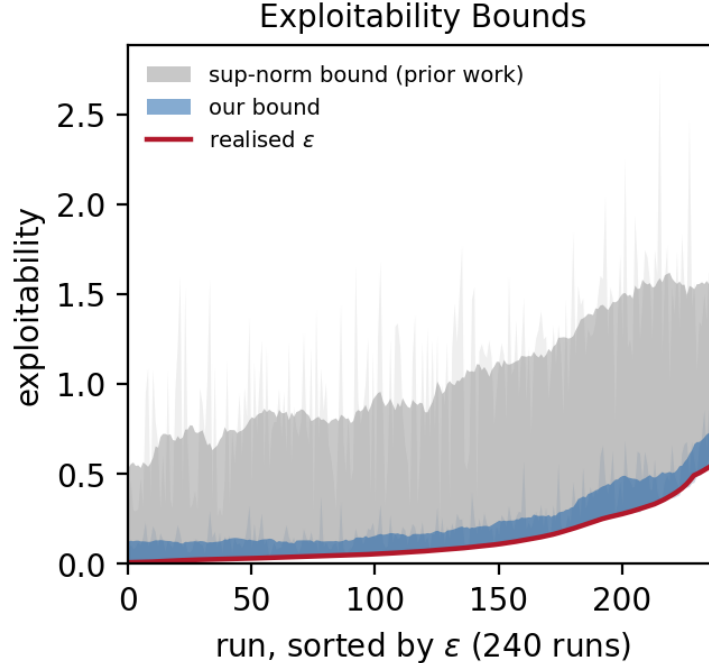


Figure 5: Realised exploitability and its bounds on the estimator’s own output, over all 240 fixed-pool budget-sweep runs on the three 21×21 games (the informative and arbitrary-embedding latent-quality games and Colonel Blotto; budgets $\{5, 10, 20, 40, 70\}\%$ of the matrix; 16 seeds each), sorted by ϵ . Shaded regions are each bound’s margin over ϵ .

References

- [1] Abdullah Al-Dujaili, Erik Hemberg, and Una-May O’Reilly. Approximating Nash equilibria for black-box games: A Bayesian optimization approach. In *Workshop on Optimization in Multiagent Systems (OptMAS)*, 2018. URL <https://arxiv.org/abs/1804.10586>.
- [2] Alex Andrew, Sam Spillard, Joshua Collyer, and Neil Dhir. Developing optimal causal cyber-defence agents via cyber security simulation. *arXiv preprint arXiv:2207.12355*, 2022.
- [3] Enrique Areyan Viqueira, Amy Greenwald, Cyrus Cousins, and Eli Upfal. Learning simulation-based games from data. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1778–1780, 2019.
- [4] Enrique Areyan Viqueira, Cyrus Cousins, and Amy Greenwald. Improved algorithms for learning equilibria in simulation-based games. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 79–87, 2020.
- [5] David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech M. Czarnecki, Julien Pérolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. In *International Conference on Machine Learning (ICML)*, 2019.
- [6] Ronen I. Brafman and Moshe Tennenholtz. R-max: A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

- [7] George W. Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 1951.
- [8] Cyrus Cousins, Bhaskar Mishra, Enrique Areyan Viqueira, and Amy Greenwald. Computational and data requirements for learning generic properties of simulation-based games. *arXiv preprint arXiv:2208.06400*, 2022.
- [9] Wojciech M. Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. Real world games look like spinning tops. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [10] Harry Emerson, Liz Zhou, David Bowman, et al. CybORG++: An enhanced gym for the development of autonomous cyber agents. *arXiv preprint arXiv:2410.16324*, 2024.
- [11] Kim Hammar. CSLE: A reinforcement learning platform for autonomous security management. In *Proceedings of the Ninth Annual Conference on Machine Learning and Systems (MLSys)*, 2026.
- [12] Kim Hammar and Rolf Stadler. Finding effective security strategies through reinforcement learning and self-play. In *Proceedings of the 16th International Conference on Network and Service Management (CNSM)*, pages 1–9. IEEE, 2020.
- [13] Minbiao Han, Fengxue Zhang, and Yuxin Chen. No-regret learning of Nash equilibrium for black-box games via Gaussian processes. In *Proceedings of the 40th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2024. URL <https://arxiv.org/abs/2405.08318>.
- [14] Steven R. Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 49(2):1055–1080, 2021.
- [15] Patrick R. Jordan, Yevgeniy Vorobeychik, and Michael P. Wellman. Searching for approximate equilibria in empirical games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1063–1070, 2008.
- [16] Thomas Kunz, Christian Fisher, James La Novara-Gsell, Christopher Nguyen, and Li Li. A multiagent CyberBattleSim for RL cyber operation agents. In *Proceedings of the International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 897–903. IEEE, 2022.
- [17] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [18] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [19] Michael Lanier and Yevgeniy Vorobeychik. CyGym: A simulation-based game-theoretic analysis framework for cybersecurity. In *Proceedings of GameSec*, LNCS 16223. Springer, 2026.
- [20] Michael Lanier and Yevgeniy Vorobeychik. A scalable approach to solving simulation-based network security games. In *Proceedings of FLAIRS*, 2026.

- [21] H. Brendan McMahan, Geoffrey J. Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *International Conference on Machine Learning (ICML)*, 2003.
- [22] Microsoft Defender Research Team. CyberBattleSim. <https://github.com/microsoft/cyberbattlesim>, 2021.
- [23] Bhaskar Mishra, Cyrus Cousins, and Amy Greenwald. Regret pruning for learning equilibria in simulation-based games. *arXiv preprint arXiv:2211.16670*, 2022.
- [24] Andres Molina-Markham, Cory Minitier, Becky Powell, and Ahmad Ridley. Network environment design for autonomous cyberdefense. *arXiv preprint arXiv:2103.07583*, 2021.
- [25] Austin A. Nguyen and Michael P. Wellman. Conservative equilibrium discovery in offline game-theoretic multiagent reinforcement learning. In *Proceedings of the 25th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2026. URL <https://arxiv.org/abs/2603.00374>.
- [26] Sean Oesch, Amul Chaulagain, Brian Weber, Matthew Dixson, Amir Sadovnik, Benjamin Roberson, Cory Watson, and Phillipe Austria. Towards a high fidelity training environment for autonomous cyber defense agents. In *Proceedings of the 17th Cyber Security Experimentation and Test Workshop, CSET '24*, pages 91–99. Association for Computing Machinery, 2024. doi: 10.1145/3675741.3675752.
- [27] Shayegan Omidshafiei, Christos Papadimitriou, Georgios Piliouras, Karl Tuyls, Mark Rowland, Jean-Baptiste Lespiau, Wojciech M. Czarnecki, Marc Lanctot, Julien Pérolat, and Rémi Munos. α -rank: Multi-agent evaluation by evolution. *Scientific Reports*, 9, 2019.
- [28] Victor Picheny, Mickael Binois, and Abderrahmane Habbal. A Bayesian optimization approach to find Nash equilibria. *Journal of Global Optimization*, 73(1):171–192, 2019. doi: 10.1007/s10898-018-0688-0.
- [29] Tabish Rashid, Cheng Zhang, and Kamil Ciosek. Estimating α -rank by maximizing information gain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5673–5681, 2021. doi: 10.1609/aaai.v35i6.16712.
- [30] Mark Rowland, Shayegan Omidshafiei, Karl Tuyls, Julien Pérolat, Michal Valko, Georgios Piliouras, and Rémi Munos. Multiagent evaluation under incomplete information. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [31] Rahul Savani and Theodore L. Turocy. Gambit: The package for computation in game theory, version 16.2.1. <https://www.gambit-project.org>, 2025.
- [32] Jonathon Schwartz and Hanna Kurniawati. Autonomous penetration testing using reinforcement learning. *arXiv:1905.05965*, 2019.
- [33] Burr Settles. Active learning literature survey. *University of Wisconsin-Madison Department of Computer Sciences Technical Report 1648*, 2009.
- [34] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2016.

- [35] Samuel Sokota, Caleb Ho, and Bryce Wiedenbeck. Learning deviation payoffs in simulation-based games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.
- [36] Niranjana Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.
- [37] Maxwell Standen, Martin Lucas, David Bowman, Toby J. Richer, Junae Kim, and Damian Marriott. CybORG: A gym for the development of autonomous cyber agents. In *IJCAI-21 1st International Workshop on Adaptive Cyber Defense*, 2021.
- [38] TTCP CAGE Working Group. Cyber operations research gym. <https://github.com/cage-challenge/CybORG>, 2022.
- [39] Karl Tuyls, Julien Pérolat, Marc Lanctot, Joel Z. Leibo, and Thore Graepel. A generalised method for empirical game theoretic analysis. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018.
- [40] Karl Tuyls, Julien Pérolat, Marc Lanctot, Edward Hughes, Richard Everett, Joel Z. Leibo, Csaba Szepesvári, and Thore Graepel. Bounds and dynamics for empirical game theoretic analysis. *Autonomous Agents and Multi-Agent Systems*, 34(7), 2020.
- [41] Yevgeniy Vorobeychik and Michael P. Wellman. Stochastic search methods for Nash equilibrium approximation in simulation-based games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1055–1062, 2008.
- [42] Yevgeniy Vorobeychik, Michael P. Wellman, and Satinder Singh. Learning payoff functions in infinite games. *Machine Learning*, 67(1–2):145–168, 2007.
- [43] Yongzhao Wang, Qiurui Ma, and Michael P. Wellman. Evaluating strategy exploration in empirical game-theoretic analysis. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1346–1354, 2022.
- [44] Yongzhao Wang, Qiurui Ma, and Michael P. Wellman. Empirical game-theoretic analysis for mean field games. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1025–1033, 2023.
- [45] Michael P. Wellman. Methods for empirical game-theoretic analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2006.
- [46] Michael P. Wellman, Karl Tuyls, and Amy Greenwald. Empirical game-theoretic analysis: A survey. *Journal of Artificial Intelligence Research*, 2025. arXiv:2403.04018.
- [47] Bryce Wiedenbeck and Michael P. Wellman. Scaling simulation-based game analysis through deviation-preserving reduction. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 931–938, 2012.
- [48] Bryce Wiedenbeck, Ben-Alexander Cassell, and Michael P. Wellman. Bootstrap statistics for empirical games. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 597–604, 2014.

A Appendix

This appendix collects the proofs, the specification of the ANSG domain, implementation details and hyperparameters, and additional experiments.

A.1 Proofs

This appendix collects full proofs of the results in Section 4, in order of appearance. Throughout, rows index defender strategies $d_1, \dots, d_{n_D}$ and columns attacker strategies $a_1, \dots, a_{n_A}$; $D, \hat{D} \in \mathbb{R}^{n_D \times n_A}$ hold the defender's true and estimated payoffs, and $A, \hat{A} \in \mathbb{R}^{n_A \times n_D}$ the attacker's, indexed attacker-first. For a mixed profile (p, q) the defender's regret in a matrix M is

$$\text{reg}_D(p, q; M) = \max_i e_i^\top M q - p^\top M q,$$

and symmetrically $\text{reg}_A(p, q; N) = \max_j e_j^\top N p - q^\top N p$ for the attacker. A profile is an ε -Nash equilibrium of (D, A) when both regrets are at most ε , and its exploitability is $\varepsilon = \max\{\text{reg}_D, \text{reg}_A\}$. That $(\hat{p}, \hat{q})$ is an $\varepsilon_{\text{solve}}$ -approximate equilibrium of $(\hat{D}, \hat{A})$ means

$$\max_i e_i^\top \hat{D} \hat{q} \leq \hat{p}^\top \hat{D} \hat{q} + \varepsilon_{\text{solve}}, \quad \max_j e_j^\top \hat{A} \hat{p} \leq \hat{q}^\top \hat{A} \hat{p} + \varepsilon_{\text{solve}}. \quad (5)$$

Write $\Delta_D = \hat{D} - D$ and $\Delta_A = \hat{A} - A$, and $|\cdot|$ for the entrywise absolute value.

Proof of Lemma 1. We prove the four links of Eq. (3) for the defender in turn, then the attacker's analogue and the consequence for ε .

We begin with form (a). Fix a pure deviation i . Since $\hat{D} = D + \Delta_D$,

$$e_i^\top D \hat{q} = e_i^\top \hat{D} \hat{q} - e_i^\top \Delta_D \hat{q}.$$

By Eq. (5), $e_i^\top \hat{D} \hat{q} \leq \hat{p}^\top \hat{D} \hat{q} + \varepsilon_{\text{solve}}$, and expanding $\hat{D}$ once more, $\hat{p}^\top \hat{D} \hat{q} = \hat{p}^\top D \hat{q} + \hat{p}^\top \Delta_D \hat{q}$. Combining,

$$e_i^\top D \hat{q} - \hat{p}^\top D \hat{q} \leq \varepsilon_{\text{solve}} + \hat{p}^\top \Delta_D \hat{q} + (-\Delta_D \hat{q})_i.$$

The right-hand side depends on i only through the last term, so taking the maximum over i on both sides gives $\text{reg}_D - \varepsilon_{\text{solve}} \leq \max_i (-\Delta_D \hat{q})_i + \hat{p}^\top \Delta_D \hat{q}$, which is (a). No property of the game other than bilinearity has been used; in particular the argument does not assume zero sum.

To pass from (a) to (b), we bound the two terms separately. For every row i , $(-\Delta_D \hat{q})_i = \sum_j \hat{q}_j (-\Delta_{D,ij}) \leq \sum_j \hat{q}_j |\Delta_{D,ij}| = (|\Delta_D| \hat{q})_i$, because $\hat{q}_j \geq 0$ and $-x \leq |x|$; taking the maximum over i gives $\max_i (-\Delta_D \hat{q})_i \leq \tau_D(\hat{q})$. Likewise $\hat{p}^\top \Delta_D \hat{q} = \sum_{i,j} \hat{p}_i \hat{q}_j \Delta_{D,ij} \leq \sum_{i,j} \hat{p}_i \hat{q}_j |\Delta_{D,ij}| = \hat{p}^\top |\Delta_D| \hat{q}$, since all weights are non-negative.

To pass from (b) to (c), write $\hat{p}^\top |\Delta_D| \hat{q} = \sum_i \hat{p}_i (|\Delta_D| \hat{q})_i$. This is a convex combination of the numbers $(|\Delta_D| \hat{q})_i$, since $\hat{p}$ is a probability vector, and a convex combination never exceeds its largest element. Hence $\hat{p}^\top |\Delta_D| \hat{q} \leq \max_i (|\Delta_D| \hat{q})_i = \tau_D(\hat{q})$, and (b) $\leq 2\tau_D(\hat{q})$.

Finally, for (c) to (d), note that for every row, $(|\Delta_D| \hat{q})_i = \sum_j \hat{q}_j |\Delta_{D,ij}| \leq \|\Delta_D\|_\infty \sum_j \hat{q}_j = \|\Delta_D\|_\infty \leq \tau$, so $\tau_D(\hat{q}) \leq \tau$. The sum ranges only over $j \in \text{supp}(\hat{q})$: an entry in a column of zero mass contributes nothing, however large. For the equality condition, suppose $\tau_D(\hat{q}) = \tau$. Then some row i attains $\sum_j \hat{q}_j |\Delta_{D,ij}| = \tau$; since each term satisfies $|\Delta_{D,ij}| \leq \tau$ and the weights sum to one, this forces $|\Delta_{D,ij}| = \tau$ for every j with $\hat{q}_j > 0$. So a worst-error cell lies in a played column, which is the stated necessary condition.

The same argument covers the attacker, whose payoff to pure strategy j against $\hat{p}$ is $e_j^\top \hat{A} \hat{p}$; the four steps apply with the roles exchanged: replace $(D, \hat{D}, \hat{p}, \hat{q}, i)$ by $(A, \hat{A}, \hat{q}, \hat{p}, j)$ and use the second

inequality of Eq. (5). This gives the attacker's chain with $\tau_A(\hat{p}) = \max_j(|\Delta_A|\hat{p})_j$ in place of $\tau_D(\hat{q})$; played rows now play the role of played columns.

Each form bounds its player's regret minus $\varepsilon_{\text{solve}}$. The exploitability is the larger of the two regrets, so it is at most $\varepsilon_{\text{solve}}$ plus the larger of the two players' values of whichever form is used, which makes $(\hat{p}, \hat{q})$ an ε -Nash equilibrium of (D, A) for that ε . $\square$

Proof of Theorem 1. The argument has three steps: a confidence event under which every simulated entry is accurate, a split of each row's weighted error on that event, and an application of Lemma 1.

We first build a confidence event under which every simulated entry is accurate. Fix one payoff entry, say D_{ij} , and let $\bar{X}_n$ be the average of its first n rollouts, each equal to D_{ij} plus independent σ -sub-Gaussian noise. A stitched time-uniform boundary of the kind developed by Howard et al. [14] gives, for any $\delta' \in (0, 1)$,

$$\Pr \left[\exists n \geq 1 : \left| \bar{X}_n - D_{ij} \right| > \sigma \sqrt{\frac{2(\log(2/\delta') + \log \log_2(2n))}{n}} \right] \leq \delta',$$

where the factor 2 inside the logarithm accounts for the two tails. The guarantee holds for all n simultaneously, and therefore also at any random visit count, including one chosen by an acquisition rule on the basis of earlier samples; this is what makes it valid under adaptive sampling, where a fixed- n bound would not be. There are $n_D n_A$ entries in D and $n_D n_A$ in A . Setting $\delta' = \delta/(2n_D n_A)$ and taking a union bound, the event

$$\mathcal{E} : \left| \hat{D}_{ij} - D_{ij} \right| \leq \zeta(m_{ij}, \delta) \text{ and } \left| \hat{A}_{ji} - A_{ji} \right| \leq \zeta(m_{ij}, \delta) \\ \text{for every simulated cell } (i, j)$$

holds with probability at least $1 - \delta$, where m_{ij} is the cell's visit count and $\log(2/\delta') = \log(4n_D n_A/\delta)$ as in ζ . Finally, $\zeta(\cdot, \delta)$ is non-increasing in the visit count: the numerator grows only doubly logarithmically, while the denominator grows linearly. Since every simulated cell has $m_{ij} \geq m$, on $\mathcal{E}$ every simulated entry has error at most $\zeta(m, \delta)$.

Next we split each row's weighted error on that event. Fix a row i , and let S_i and U_i be the columns whose cells in that row are simulated and surrogate-filled, respectively. On $\mathcal{E}$,

$$\begin{aligned} (|\Delta_D|\hat{q})_i &= \sum_{j \in S_i} \hat{q}_j |\Delta_{D,ij}| + \sum_{j \in U_i} \hat{q}_j |\Delta_{D,ij}| \\ &\leq \zeta(m, \delta) \sum_{j \in S_i} \hat{q}_j + \beta \sum_{j \in U_i} \hat{q}_j, \end{aligned}$$

using Step 1 on the first sum and the assumption that surrogate entries have absolute error at most β on the second. The first sum of weights is at most one. The second is the $\hat{q}$ -mass row i places on surrogate cells, which is at most its maximum over rows, $w_{\text{sur}}(\hat{q})$. Hence $(|\Delta_D|\hat{q})_i \leq \zeta(m, \delta) + \beta w_{\text{sur}}(\hat{q})$ for every i , and taking the maximum, $\tau_D(\hat{q}) \leq \zeta(m, \delta) + \beta w_{\text{sur}}(\hat{q})$. The attacker's version, $\tau_A(\hat{p}) \leq \zeta(m, \delta) + \beta w_{\text{sur}}(\hat{p})$, follows in the same way over the attacker's rows.

It remains to apply Lemma 1. Form (c) of Lemma 1 gives $\text{reg}_D \leq \varepsilon_{\text{solve}} + 2\tau_D(\hat{q})$ and $\text{reg}_A \leq \varepsilon_{\text{solve}} + 2\tau_A(\hat{p})$. Substituting Step 2 and taking the larger of the two regrets yields the stated bound on ε , on the event $\mathcal{E}$ of probability at least $1 - \delta$. Because $\mathcal{E}$ was built to hold at every visit count, the bound holds for whatever cells and visit counts the acquisition rule chose. $\square$

Proof of Corollary 1. Work on the event $\mathcal{E}$ of the previous proof, which has probability at least $1 - \delta$, and consider the defender; the attacker is symmetric.

We first bound the best deviation from above, and claim that $D \leq D^{\text{opt}} + \zeta_{m,\delta}$ entrywise. On a simulated cell, D_{ij}^{opt} is the measured mean, which on $\mathcal{E}$ is within $\zeta_{m,\delta}$ of D_{ij} . On an unsimulated cell, D_{ij}^{opt} is the maximum payoff, which bounds D_{ij} by definition. Since $\hat{q} \geq 0$ and sums to one, it follows that for every i , $e_i^\top D \hat{q} \leq e_i^\top D^{\text{opt}} \hat{q} + \zeta_{m,\delta}$, and so $\max_i e_i^\top D \hat{q} \leq \max_i (D^{\text{opt}} \hat{q})_i + \zeta_{m,\delta}$.

Next we bound the equilibrium value from below. The value term needs the matrix the solver was given to be accurate where the equilibrium plays, that is, on the support block $\text{supp}(\hat{p}) \times \text{supp}(\hat{q})$. When that block is simulated, every term of $\hat{p}^\top \hat{D} \hat{q} = \sum_{i,j} \hat{p}_i \hat{q}_j \hat{D}_{ij}$ with non-zero weight is a measured mean, so on $\mathcal{E}$, $\hat{p}^\top D \hat{q} \geq \hat{p}^\top \hat{D} \hat{q} - \zeta_{m,\delta}$.

Subtracting the second bound from the first, $\text{reg}_D = \max_i e_i^\top D \hat{q} - \hat{p}^\top D \hat{q} \leq \max_i (D^{\text{opt}} \hat{q})_i - \hat{p}^\top \hat{D} \hat{q} + 2\zeta_{m,\delta}$, which is at most the stated bound; the $\varepsilon_{\text{solve}}$ term is slack and is kept only so that the certificate dominates Theorem 1 term by term. The same argument with A^{opt} and the rows of $\text{supp}(\hat{p})$ bounds reg_A , and the exploitability is the larger of the two.

The condition on the support block cannot simply be dropped. If some support cell is surrogate-filled, the value term can overstate $\hat{p}^\top D \hat{q}$ by up to β times its weight, and that error is not visible in the data. Replacing $\hat{D}$ in the value term by a pessimistic fill, measured values on simulated cells and the minimum payoff elsewhere, restores the bound without any condition, at the price of a looser value term. $\square$

Proof of Corollary 2. We first show that both multipliers vanish. By definition,

$$w_{\text{sur}}(\hat{q}) = \max_i \sum_j \hat{q}_j \mathbf{1}[(i, j) \text{ surrogate}],$$

and only columns $j \in \text{supp}(\hat{q})$ carry weight. The hypothesis says that every cell in those columns is simulated, in every row, so for each row the sum has no non-zero term and $w_{\text{sur}}(\hat{q}) = 0$. The same argument over the rows of $\text{supp}(\hat{p})$ gives $w_{\text{sur}}(\hat{p}) = 0$.

Substituting $w_{\text{sur}} = 0$ into Theorem 1 removes the term βw_{sur} for both players, leaving $\varepsilon \leq \varepsilon_{\text{solve}} + 2\zeta(m, \delta)$. Since β appears only in the removed term, the bound holds whatever the surrogate's error.

It remains to count the cells. The set is the union of $C = \{1:n_D\} \times \text{supp}(\hat{q})$, with $n_D s_A$ cells, and $R = \text{supp}(\hat{p}) \times \{1:n_A\}$, with $n_A s_D$ cells. Their intersection is $\text{supp}(\hat{p}) \times \text{supp}(\hat{q})$, with $s_D s_A$ cells, so by inclusion and exclusion $|C \cup R| = n_D s_A + n_A s_D - s_D s_A = B^*$. $\square$

Proof of Proposition 1. Coverage is monotone: a cell once simulated is never returned to the surrogate, so the set of cached cells is non-decreasing in t .

Let $\mathcal{R}_t$ denote the set consisting of every row simulated in all n_A columns together with every column simulated in all n_D rows, after round t . Round t simulates $\{1:n_D\} \times \text{supp}(\hat{q}_t)$, which completes every column of $\text{supp}(\hat{q}_t)$, and $\text{supp}(\hat{p}_t) \times \{1:n_A\}$, which completes every row of $\text{supp}(\hat{p}_t)$; hence $\text{supp}(\hat{p}_t) \cup \text{supp}(\hat{q}_t) \subseteq \mathcal{R}_t$.

Suppose the procedure does not halt after round $t+1$, so $w_{\text{sur}}(\hat{q}_{t+1}) > 0$ or $w_{\text{sur}}(\hat{p}_{t+1}) > 0$. In the first case there are i and j with $\hat{q}_{t+1,j} > 0$ and (i, j) unsimulated, so column j is incomplete and therefore $j \notin \text{supp}(\hat{q}_t)$; round $t+1$ simulates $\{1:n_D\} \times \text{supp}(\hat{q}_{t+1}) \ni (\cdot, j)$ and so adds j to $\mathcal{R}$. The second case is symmetric in rows. Every non-halting round therefore enlarges $\mathcal{R}$ by at least one element, and $|\mathcal{R}| \leq n_D + n_A$, which gives (i).

For (ii), halting means $w_{\text{sur}}(\hat{p}) = w_{\text{sur}}(\hat{q}) = 0$ at the mixture returned by the final solve, which is exactly the hypothesis of Corollary 2 evaluated at that mixture; the conclusion of Theorem 1 then loses its β term. For (iii), monotonicity bounds the total simulated set by the whole matrix. $\square$

Proof of the per-iteration accounting above. Write $\hat{p}, \hat{q}$ for the unchanged supports and $d_{\text{new}}, a_{\text{new}}$ for the added strategies. The set $\{1:n_D\} \times \text{supp}(\hat{q})$ acquires one new row, contributing the s_A cells (d_{new}, j) for $j \in \text{supp}(\hat{q})$; the set $\text{supp}(\hat{p}) \times \{1:n_A\}$ acquires one new column, contributing the s_D cells (i, a_{new}) for $i \in \text{supp}(\hat{p})$. The cell $(d_{\text{new}}, a_{\text{new}})$ lies in neither set, since by hypothesis neither new strategy is in a support. All other cells of the coverage set were simulated at the previous iteration and are cached. A cached full rebuild instead simulates every newly exposed cell, of which there are $(k+1)^2 - k^2 = 2k+1$. $\square$

Proof of Corollary 3. Under coverage $w_{\text{sur}}(\hat{p}) = w_{\text{sur}}(\hat{q}) = 0$, so the bound of Corollary 1 reduces to $\varepsilon_{\text{solve}} + 2\zeta(m, \delta)$ and is at most ε_{tgt} exactly when $2\zeta(m, \delta) \leq \varepsilon_{\text{tgt}} - \varepsilon_{\text{solve}}$. Since $\zeta(m, \delta)$ is strictly decreasing in m , the set of admissible m is an up-set and m^* is its minimum. The cost follows because coverage simulates B^* cells and each must carry m^* visits. For the rate, drop the $\log \log_2(2m)$ term, which is $o(\log m)$: solving $\sigma \sqrt{2 \log(4n_D n_A / \delta) / m} = (\varepsilon_{\text{tgt}} - \varepsilon_{\text{solve}}) / 2$ gives $m = 8\sigma^2 \log(4n_D n_A / \delta) / (\varepsilon_{\text{tgt}} - \varepsilon_{\text{solve}})^2$, and reinstating the log log term inflates this by a factor $1 + o(1)$. $\square$

Proof of Theorem 3. For (i), coverage is monotone: a simulated cell is never returned to the surrogate. Let $\mathcal{R}$ be the set of rows simulated in every column together with the columns simulated in every row. By the argument of Proposition 1, whenever the iteration fails to halt it enlarges $\mathcal{R}$ by at least one element. Since $|\mathcal{R}| \leq n_D + n_A$ and $\mathcal{R}$ never shrinks, at most $n_D + n_A$ such extensions can occur across the whole sequence, however many times the mixture changes; after the last, every subsequent solve halts immediately and $w_{\text{sur}} = 0$ holds permanently.

For (ii), on the stated event Theorem 1 gives

$$\varepsilon \leq \varepsilon_{\text{solve}} + 2 \max_{\pi} [\zeta(m, \delta) + \beta w_{\text{sur}}(\pi)],$$

and $w_{\text{sur}} = 0$ removes the second term. The radius

$$\zeta(m, \delta) = \sigma \sqrt{\frac{2(\log(4n_D n_A / \delta) + \log \log_2(2m))}{m}}$$

is decreasing in m with $\zeta \rightarrow 0$: the numerator grows doubly logarithmically while the denominator grows linearly, so the ratio inside the root tends to 0 by L'Hôpital's rule on the ∞/∞ form. The log log term therefore costs only a bounded factor at any practical m , and ζ retains the $m^{-1/2}$ decay of a fixed-design radius; this is why the budget of Corollary 3 inherits the fixed-design rate up to that factor. Crucially the bound is time-uniform: it holds simultaneously for all m and all sample-dependent stopping rules on a single event of probability $1 - \delta$, so the limit is taken on that event without a further union bound.

For (iii), $\varepsilon_{\text{solve}}$ is the exploitability of $(\hat{p}, \hat{q})$ in the estimated game and vanishes when the restricted game is solved exactly, giving $\varepsilon \rightarrow 0$. Since ε is exploitability in the true restricted game, the limit profile is an exact equilibrium of it. The cells outside the coverage set enter neither $\max_i (Dq)_i$ nor $\max_j (Ap)_j$ by Corollary 2, so their values are unconstrained. $\square$

Proof of Lemma 2. By Eq. (2), $\varphi(\pi)$ is a function of $\{\mu_{\pi}(\xi_p)\}_{p \leq P}$ alone, so any two policies agreeing on Ξ receive identical fingerprints whatever they do elsewhere. Payoffs, by contrast, depend on behaviour at all reachable states. Choosing π' to agree with π on Ξ and to realise the desired behaviour off it, which the hypothesis permits, gives the claim. $\square$

Proof of Theorem 2. Suppose for contradiction that some such $\mathcal{A}$, at budget B , simulates a trap cell on every twin-trap instance.

Consider the *all-duplicate* instance I_0 , in which a designated attacker strategy $a_\star$ is a genuine copy of a_{twin} : same fingerprint, same payoffs, likewise weak. Run $\mathcal{A}$ on I_0 and let $\mathcal{S}_B$ be the set of cells it simulates, so $|\mathcal{S}_B| \leq B < n_D n_A$. Since $\mathcal{S}_B$ omits at least one cell, and by enlarging the pool if necessary, choose an attacker column $j_\star$ that $\mathcal{S}_B$ does not meet.

By Lemma 2 there is a policy a_{trap} with $\varphi(a_{\text{trap}}) = \varphi(a_{\text{twin}})$ whose payoffs make it a dominant deviation. Let I_1 be the instance obtained from I_0 by installing a_{trap} in column $j_\star$, leaving everything else untouched. Then I_1 is a twin-trap instance, and I_0 and I_1 differ only in the payoffs of cells in column $j_\star$.

We claim $\mathcal{A}$ executes the same run on I_0 and I_1 . We argue by induction over rounds. In the base case, before any cell is simulated, both runs hold the empty cache and the same fingerprints, since $\varphi(a_{\text{trap}}) = \varphi(a_{\text{twin}}) = \varphi(a_\star)$ and φ is a function of the actor alone, not of payoffs; the two runs therefore agree at round 1. For the inductive hypothesis, suppose the runs agree through round r , so the caches $\mathcal{C}_r$ agree as multisets of simulated cells. The entries recorded there agree as values as well: every cell in $\mathcal{C}_r \subseteq \mathcal{S}_B$ lies outside column $j_\star$, where I_0 and I_1 are identical by construction. Hence $\mathcal{A}$ receives identical inputs at round $r + 1$ and, being deterministic, selects identical cells there, so the runs agree through round $r + 1$. By induction, $\mathcal{A}$ selects the same cells on I_0 and I_1 at every round.

Therefore $\mathcal{A}$ simulates exactly $\mathcal{S}_B$ on I_1 , and $\mathcal{S}_B$ does not meet column $j_\star$, so $\mathcal{A}$ simulates no trap cell on I_1 . But I_1 is a twin-trap instance, on which the hypothesis asserts that $\mathcal{A}$ does simulate a trap cell. This is the contradiction, and it establishes the first claim.

For the exploitability claim, on I_1 the surrogate assigns column $j_\star$ its prediction for a_{twin} , since the two share a fingerprint and no label from $j_\star$ ever enters the cache. That prediction is uniformly weak, so the returned $\hat{q}$ places no mass on the trap. In the true game a_{trap} is a dominant deviation, so the attacker’s regret $\max_j (A\hat{p})_j - \hat{q}^\top A\hat{p}$ is at least $\eta := (A\hat{p})_{j_\star} - \hat{q}^\top A\hat{p}$, the gap between the trap’s true payoff against $\hat{p}$ and the value the estimated game reports. This is fixed by the instance and does not shrink with B .

Uniform draws escape the argument because the construction needs a deterministic $\mathcal{S}_B$ outside of which to place the trap. A rule visiting each unsimulated cell with probability bounded below admits no such placement, and reaches column $j_\star$ almost surely. $\square$

A.2 Fingerprint collisions

Lemma 2 (Fingerprint collisions exist). *Fix a probe bank $\Xi = \{\xi_1, \dots, \xi_P\}$ and let Π be a policy class in which, for any $\pi \in \Pi$ and any prescribed behaviour off Ξ , some $\pi' \in \Pi$ agrees with π on Ξ and realises that behaviour elsewhere. Then for every $\pi \in \Pi$ there is $\pi' \in \Pi$ with $\varphi(\pi') = \varphi(\pi)$ whose payoffs against a fixed opponent are arbitrary.*

The hypothesis on Π is mild. It asks only that agreement on P probe states does not determine behaviour everywhere, which fails only if Ξ happens to be a sufficient statistic for the policy class. For neural controllers on a state space larger than the probe bank, it holds.

A.3 Extended discussion

This appendix gives the longer treatment of three topics the main text summarises: how RWPS relates to prior payoff-estimation and equilibrium-search methods, what the coverage result adds to known EGTA principles and how it prices a certificate, and further detail on the growing-pool PSRO comparison. Throughout, RWPS is the budgeted estimator of Algorithm 1: it simulates a subset of payoff cells, fills the rest with an ensemble surrogate over strategy embeddings, and chooses cells by a score built from bootstrap equilibrium marginals. The *deviation-relevant set* is the set

of cells pairing every strategy of one player with each strategy in the support of the other player’s equilibrium mixture.

A.3.1 Relation to prior payoff-estimation methods

GS, PSP and PS-REG-M allocate samples so that a uniform ε -approximation guarantee holds over every cell, pruning profiles once their bounds separate. We run all three with the sampling schedules from their published works. At the budgets RWPS targets that guarantee cannot be bought: PSP’s published schedule begins at 294 samples per cell, whereas a 5% per-build budget on a 14×14 pool affords ten cell evaluations in total. None of the three methods therefore prunes a single profile, and all return the history-only estimate. This reflects the operating regime, not a flaw in those algorithms, which are designed for budgets that cover the matrix many times over.

Sokota et al. [35] also pair a learned model with an acquisition rule defined at the level of the solution. They train a network mapping mixed profiles to deviation payoffs and refine it by sampling in the neighbourhood of each candidate equilibrium. Three differences separate their setting from ours. Their model takes a mixture as input and returns deviation payoffs directly, so no payoff table is built, whereas RWPS estimates individual cells over strategy embeddings and passes the completed matrix to a solver. They exploit role symmetry to scale in the number of players, while our games are two-player, asymmetric, and have pools that grow by best response. And their refinement is validated empirically, with no bound on the resulting equilibrium, which is what Theorem 1 and Corollary 2 supply. Because their rule is a score over cells rather than a full payoff-build method, the fair comparison runs it inside an otherwise identical build, which is how Table 3 reports it.

Choosing which expensive evaluation to run is the subject of active learning and Bayesian optimisation [33, 34], and deep ensembles [17] supply the per-cell uncertainty RWPS uses. Picheny et al. [28] apply Bayesian optimisation to Nash equilibrium computation in black-box games, treating the equilibrium rather than a payoff as the object of interest. They target continuous strategy spaces with a Gaussian-process surrogate and a fixed game; RWPS targets a finite pool that grows across PSRO iterations and reuses every simulated entry as training data. Both depart from standard Bayesian optimisation in the target: accuracy of the equilibrium of the whole matrix, not pointwise accuracy. RWPS measures uncertainty about that equilibrium by re-solving bootstrap realisations of the game, which is nearly free because solving is cheap next to simulating.

Of their two acquisition criteria, probability of equilibrium has a discrete analogue, reported as the bootstrap-inclusion variant in Appendix A.9.3: a cell is scored by how often it enters a bootstrap support instead of by a Gaussian-process posterior. Stepwise uncertainty reduction does not carry over. It targets the probability that a cell *is* the equilibrium, which is defined for the pure equilibria of a discretised space but not for the mixed equilibria, with supports (2, 2) and (12, 12), that our games have. Extending it to mixed equilibria is open.

Two lines extend Picheny et al. [28]. Al-Dujaili et al. [1] search for pure equilibria of continuous black-box games by Gaussian-process regression on an approximate regret surface, with an ε -greedy trade-off between exploiting that surface and reducing uncertainty. Han et al. [13] give ARISE, which restricts acquisition to a region of interest around the current equilibrium estimate and proves no-regret guarantees for the resulting query sequence. Both minimise a global regret functional over a continuous strategy space and query strategy profiles directly. RWPS instead completes a finite payoff matrix whose rows and columns are policies produced by PSRO, reuses every cell across iterations, and certifies the equilibrium of the completed matrix; ARISE’s region-of-interest restriction plays a role similar to our equilibrium weighting, and its no-regret analysis is over queries rather than over the returned mixture.

Nguyen and Wellman [25] also face payoffs that cannot be freely simulated, but from the opposite

direction: their dataset is fixed, so they replace the simulator with an ensemble dynamics model and steer best responses away from profiles the data does not support. Our setting keeps the simulator and rations calls to it, so uncertainty drives *which* cells to pay for rather than a conservatism penalty on responses. The two are complementary: a budgeted estimator decides what to simulate next, and a conservative response objective decides what to trust when nothing more can be simulated.

Rowland et al. [30] maintain confidence intervals over payoffs until the response graph is determined, and Rashid et al. [29] choose each query to maximise information gain about α -rank. Neither optimises exploitability against a Nash mixture, so neither is run as a baseline. Information-gain acquisition is represented among the baselines through the IGS method of Jordan et al. [15].

A.3.2 Coverage, budgets and certificates

That verifying a profile’s regret requires only the profile and its unilateral deviations is long established in empirical game-theoretic analysis (EGTA) [45]. Wellman et al. [46] state it directly and note that the count stays modest for mixed profiles with small supports; the same idea underlies minimum-regret-first search (MRFS) [15] and deviation-preserving reduction [47]. We do not claim the principle. Corollary 2 adds two things: a closed-form count, $B^* = n_D s_A + n_A s_D - s_D s_A$ cells for a bimatrix game with pool sizes n_D, n_A and support sizes s_D, s_A , and a coupling to the error bound of Theorem 1.

The coupling is the more consequential part. In prior treatments the cells outside the deviation set are *unevaluated*: pruned, deferred or unknown, with guarantees stated only over what was sampled. In RWPS they are *filled* by a surrogate the solver reads, and the corollary shows that however inaccurate that fill is, it cannot reach either player’s regret; the surrogate term w_{sur} is zero regardless of the surrogate’s error β . This is what makes interpolation admissible in place of pruning, and what turns the deviation-relevant set into a budget for an estimator rather than a stopping rule for a sampler.

The connection is exact. MRFS confirms a *pure* profile of a two-player $n \times n$ game by evaluating its entire unilateral deviation set, the row and column through that cell, which is $2n - 1$ evaluations. Setting $s_D = s_A = 1$ in Corollary 2 gives $n + n - 1 = 2n - 1$. The deviation-relevant set is MRFS’s confirmation criterion extended from a pure profile to a mixed one.

We need full columns on the opponent’s support, since only those columns enter a player’s expected payoff, and full rows on a player’s own support, since deviations range over all of that player’s strategies. Nothing outside this set affects either player’s regret. Coverage does not simulate the whole matrix: the remaining $n_D n_A - B^*$ cells stay surrogate-filled and are read by the solver, and the corollary shows only that they carry zero weight in regret.

Theorem 1 bounds exploitability by a statistical term, from finite rollouts per simulated cell, and a representational term β , from surrogate error on unsimulated cells. Only the statistical term shrinks with spending. The representational term depends on whether fingerprints happen to predict payoffs in the game at hand, cannot be certified in advance, and by Theorem 2 can be arbitrarily large. Covering the deviation-relevant set removes it outright, leaving a purely statistical certificate: the guarantee a full rebuild would give, without a full rebuild and without any assumption about surrogate quality.

B^* is linear in pool and support sizes, so support size rather than matrix size sets the price of coverage. The support sizes are unknown before solving, but they can be estimated by re-solving many bootstrap realisations of the surrogate-filled game, following the bootstrap methodology of Wiedenbeck et al. [48]; this costs solves, not rollouts. Before committing any simulation, a practitioner can therefore read off whether coverage is a small fraction of the matrix, in which case budgeted estimation is worth enabling, or close to all of it, in which case a full rebuild is the better

choice. Treated as a one-shot prediction the estimate can be wrong; Appendix A.8 shows how to iterate it to a fixed point instead.

Coverage fixes *which* cells are simulated; the number of rollouts per cell fixes *how tight* the certificate of Corollary 1 is.

Corollary 3 (Budget for a target certificate). *Fix δ and a target $\varepsilon_{\text{tgt}} > \varepsilon_{\text{solve}}$. If the deviation-relevant set is covered, so that $w_{\text{sur}} = 0$, then the certificate of Corollary 1 is at most ε_{tgt} as soon as every simulated cell carries*

$$m \geq m^* = \min\{m : 2\zeta(m, \delta) \leq \varepsilon_{\text{tgt}} - \varepsilon_{\text{solve}}\},$$

and the total rollout cost of achieving it is $B^ \cdot m^*$, with*

$$m^* = O\left(\frac{\sigma^2 \log(n_D n_A / \delta)}{(\varepsilon_{\text{tgt}} - \varepsilon_{\text{solve}})^2}\right)$$

up to the log log factor carried by the time-uniform radius.

Here $\zeta(m, \delta)$ is the concentration radius for a cell mean from m rollouts, σ the payoff noise scale, and $\varepsilon_{\text{solve}}$ the solver’s own regret. The two factors are bought separately and neither substitutes for the other: B^* removes β , and m^* shrinks ζ . Both are computable in advance, B^* from bootstrap supports and m^* by inverting ζ , and every quantity in the certificate is computable from simulation data. A deployment can therefore quote a budget against a target exploitability before spending it, including where the true equilibrium is unavailable. Coverage is cheap and precision is not: at $\sigma = 0.1$ on a 30×30 pool, 2ζ is roughly half the payoff range at $m = 4$, and reaching 0.05 needs $m^* \approx 512$, while pool size enters only logarithmically.

The unsigned forms of Eq. (3) charge errors in both directions, but the signed form (a) does not: only a deviation row whose value the estimate understates, or a support whose value it overstates, produces regret. This is what makes optimism admissible. Filling the unsimulated cells of a player’s own matrix at the maximum payoff, in the tradition of R-max [6], understates no deviation, so the deviation term reduces to statistical noise on simulated cells and only the support term remains; Corollary 1 uses exactly this fill, and only to evaluate the bound after the solve. Using the same fill inside the solve instead changes the object solved. Every unsimulated cell then looks maximal, so a best response points at it; it enters the support, is simulated, proves ordinary, and leaves, and the next unsimulated cell takes its place. With $n_D n_A$ cells to chase, this certifies only after nearly the whole matrix is bought (Appendix A.10). Optimism certifies a solution; it does not save simulation.

A.3.3 Necessity of forced exploration

The asymptotic guarantee of RWPS rests on its forced-exploration draws. They visit every cell infinitely often almost surely, so running means converge and Lemma 1 with $\tau_{\text{supp}} \rightarrow 0$ closes the argument; the two-phase Borel–Cantelli proof is standard. No acquisition rule that reads only fingerprints and cached payoffs can replace these draws. A fingerprint is a finite summary of a policy’s behaviour, so distinct policies can share one while differing arbitrarily in payoff (Appendix A.2), and such a rule cannot tell them apart.

Theorem 2 (Necessity of exploration). *Call an attacker strategy a_{twin} a twin if it is densely simulated and uniformly weak, and a_{trap} a trap if $\varphi(a_{\text{trap}}) = \varphi(a_{\text{twin}})$ while its true payoffs make it a dominant deviation. Let $\mathcal{A}$ be any deterministic acquisition rule whose selections are a function of the fingerprints $\{\varphi(s)\}_{s \in S_D \cup S_A}$ and the cache $\mathcal{C}$ alone. Then for every budget $B < n_D n_A$ there is a twin–trap instance on which $\mathcal{A}$ simulates no trap cell, and on which the returned mixture has exploitability at least $\eta > 0$ with η determined by the instance and independent of B .*

A.3.4 Progressive sampling at budgets where it prunes

The growing-pool comparison runs the progressive-sampling family below the budgets it was designed for. To show where it does become competitive, Table 2 in the main paper, and Table 4 in full, compare RWPS with GS, PSP and PS-REG-M on fixed 21×21 pools at budgets from $0.2\times$ to $100\times$ the 441-cell matrix, with each family member implemented from its original specification. PS-REG-M, the member with mixed-equilibrium guarantees, prunes nothing below $100\times$; there it prunes 0.5% of utility indices on the informative game and 6.1% on Blotto. PSP prunes nothing at any budget tested, and the pure-equilibrium variant PS-REG+ starts earlier, pruning 0.7% and 4.2% at $20\times$. At $100\times$ PSP and PS-REG-M reach exploitability comparable to RWPS: none of the four differences is significant (paired t -test, $p \geq 0.15$). Below that, GS and PSP coincide exactly at $0.2\times$, because neither can certify any cell and both fall back to a flat fill. Across all thirty pairwise comparisons RWPS is significantly better in twenty and never significantly worse ($p < 0.05$). The comparison is therefore one of operating regime: the family buys a uniform PAC guarantee and pays its sample complexity, which is affordable only at budgets many times the matrix. This sweep predates the final acquisition rule and uses four seeds, so we report it as a regime check rather than as a head-to-head result.

Table 4: All five budgets of the progressive-sampling comparison (extends Table 2); exploitability mean $\pm$ SD over four seeds, bold marks each row’s best.

Game	Budget	RWPS	GS	PSP	PS-REG-M	pruned
Inf.	$0.2\times$	.051 $\pm$.013	.307 $\pm$.112	.307 $\pm$.112	.166 $\pm$.136	0
	$1\times$	.029 $\pm$.023	.162 $\pm$.138	.222 $\pm$.081	.214 $\pm$.106	0
	$5\times$	.037 $\pm$.019	.230 $\pm$.091	.354 $\pm$.195	.546 $\pm$.331	0
	$20\times$	.012 $\pm$.006	.413 $\pm$.125	.156 $\pm$.298	.029 $\pm$.050	0
	$100\times$	.007 $\pm$.003	.025 $\pm$.040	.005 $\pm$.003	.003 $\pm$.002	0.5%
Blotto	$0.2\times$	.135 $\pm$.045	.488 $\pm$.024	.488 $\pm$.024	.501 $\pm$.041	0
	$1\times$	.043 $\pm$.011	.475 $\pm$.029	.471 $\pm$.026	.617 $\pm$.268	0
	$5\times$	.032 $\pm$.018	.556 $\pm$.093	.419 $\pm$.050	.440 $\pm$.080	0
	$20\times$	.025 $\pm$.004	.561 $\pm$.081	.366 $\pm$.081	.404 $\pm$.049	0
	$100\times$	.015 $\pm$.003	.440 $\pm$.035	.013 $\pm$.001	.017 $\pm$.007	6.1%

A.3.5 Growing-pool PSRO: further detail

In the growing-pool comparison of Table 1, every method starts from the same pool, but each method’s best-response oracle trains against that method’s own restricted equilibrium, so a better payoff estimate steers PSRO toward different strategies and the pools diverge after the first iteration. Exploitability is measured against the full game by zero-lifting each restricted mixture, which keeps pools of different composition comparable. The comparison is therefore end-to-end: it credits RWPS both for its payoff estimates and for the strategies those estimates lead PSRO to find. Appendix A.7 repeats it with identical payoffs and uninformative fingerprints to isolate what the fingerprint contributes.

RWPS starts behind: at iteration 0 its exploitability is 0.262, against 0.180 for MRFS and 0.208 for uniform sampling. This is expected. The cache is empty at the first build, so the surrogate is a rank-one fit to almost no labels and there is no history to reuse. The advantage of RWPS lies in what it carries between builds, and its lead widens monotonically with the iteration index.

IGS finishes behind both MRFS and RWPS on both games (Table 1). Jordan et al. [15]

recommend IGS for noisy payoffs, where MRFS is their method for exactly revealed payoffs, and report it outperforming the ECVI benchmark. We implement it from their Section 6.2: the next profile sampled is the one maximising Kullback–Leibler information gain aggregated over its deviation set, with the minimum-regret probability of each profile taken from their point approximation, and an MRFS prefix supplying initial samples as in their IGS-MRFS-3 variant.

The budget, not the algorithm, is what puts IGS behind. IGS reallocates samples among profiles it has already visited; its original experiments run at roughly twenty-eight samples per profile on a thirty-five-profile game. A budget that cannot visit every profile once denies it the quantity it reasons about. On a fixed 21×21 pool, IGS improves from 0.177 to 0.041 as its allowance grows from 0.2 to 1.0 evaluations per profile, converging toward MRFS.

Figure 6 reports an earlier CyGym comparison of RWPS and uniform sampling as a function of cumulative simulator calls.

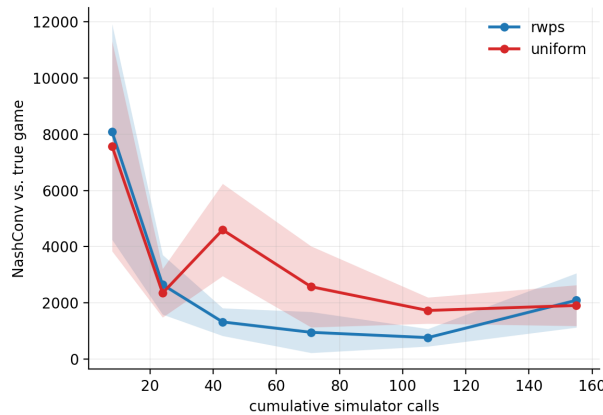


Figure 6: NashConv against cumulative simulator calls in CyGym (lower is better). At the lowest and highest budgets uniform sampling suffices; across all budget levels RWPS meets or exceeds it.

The bound evaluated in Figure 5 includes the solver’s own regret $\varepsilon_{\text{solve}}$. Without that term the signed bound is violated on three runs, all on Blotto and each by less than that run’s solver regret. Fictitious play converges slowly on games with large supports, and raising its iteration count from 400 to 20,000 does not remove the residual: measured $\varepsilon_{\text{solve}}$ averages 0.008 on Blotto against 0.002 on the informative game. Because $\varepsilon_{\text{solve}}$ is computable from the estimated matrices alone, including it costs nothing in applicability and restores validity on every run, at a cost of 0.006 in mean tightness.

A.3.6 Cyber-defence simulators

RWPS needs a simulator in which both an attacker and a defender are learning players, since PSRO grows a policy pool for each side and every payoff entry pairs one policy from each. We surveyed the environments in current use against that requirement; Table 5 summarises the result.

CybORG [37], which underpins the CAGE challenges [38], and CyGym [19] both support learning agents on each side, and we use the latter directly. Cyberwheel [26] supports one learning red and one learning blue agent simultaneously and is the third simulator in our evaluation. gym-idsgame [12] models intrusion prevention as a Markov game between a learned attacker and a learned defender, and its accompanying study trains both by self-play against an opponent pool. It is the closest prior use of a policy pool in this domain, though it estimates no payoff matrix and so has no counterpart to our estimator. The multiagent extension of CyberBattleSim [16] adds a trainable defender and

reports joint training of red and blue policies, which makes it a candidate once its code is publicly maintained.

Several widely used environments do not meet the requirement. In YAWNING-TITAN [2] only the defender learns and the attacker is scripted and probabilistic, and in the standard CyberBattleSim [22] the defender is a predefined stochastic process. NASim [32] models a penetration-testing attacker with no active defender. CSLE [11] is a broad platform combining emulation and simulation across several security-management use cases rather than a single two-player environment; we have not identified a configuration of it that fits our setting.

Table 5: Cyber-defence simulators by whether each side is a learning player. NASim has no defender at all.

Simulator	Learning red	Learning blue	Role here
CyGym	✓	✓	evaluated
Cyberwheel	✓	✓	evaluated
CybORG / CAGE	✓	✓	related
gym-idsgame	✓	✓	related
Multiagent CyberBattleSim	✓	✓	candidate
YAWNING-TITAN	✗	✓	excluded
CyberBattleSim	✓	✗	excluded
NASim	✓	✗	excluded
CSLE	platform		deferred

A.4 The Agentic Network Security Game

This appendix gives the full specification of the ANSG introduced in Section 5.

CyGym [19] models compromise as binary node access: the attacker exploits a node and thereby controls it. In a network of LLM agents the relevant failure is different. An agent processes workloads by invoking tools, and an adversary who plants instructions in content the agent reads can cause it to execute attacker-chosen goals *in addition to* the ones its principal asked for. The legitimate task still completes, so the subversion need not look like a failure at all. ANSG replaces access-compromise with this *goal subversion* model, keeping CyGym’s binary per-node state and its PSRO solution machinery while changing what compromise means and what each player can observe.

ANSG is a two-player general-sum partially observable stochastic game

$$\mathcal{G} = \langle \mathcal{S}, O^D, O^A, \mathcal{A}^D, \mathcal{A}^A, \mathcal{T}, U^D, U^A, \gamma \rangle$$

over a directed communication graph $G = (\mathcal{V}, \mathcal{E})$ whose nodes host agents. Each node carries a behavioural mode $b_v \in \{\text{benign}, \text{malicious}\}$ and a context age h_v . Both players choose policies simultaneously without observing the other’s, so Nash equilibrium is the solution concept and PSRO the approximation method, as in CyGym.

Attacker and defender valuations are independent rather than opposed: the attacker gains the value of the subverted goals it executes, while the defender loses the value of its workloads being corrupted, and these need not sum to a constant. Per-goal value factors differ by player. This is the property that makes ANSG a natural test for a budgeted estimator whose theory assumes nothing about zero-sum structure.

The defender may *audit* a node, receiving a noisy signal whose detectability rises with the value of the subverted goal; *reprompt* it, a soft intervention with no CyGym analogue; *reset* it, which restores benign behaviour but erases context history at a cost; or *isolate* it, severing communication

edges and halting propagation. The attacker may *inject* a payload into a workload, choosing a target, a goal rank, and a channel, or *probe* to estimate which previously compromised nodes remain malicious.

Injection carries a channel choice χ . Indirect prompt injection appends an attacker goal alongside the legitimate workload, which therefore completes normally; the only signal is a rank-dependent trace. Malicious tool triggering instead redirects an existing call, denying the legitimate output for a more damaging hit at higher detection probability. The rank choice sets value against exposure, and the channel choice compounds it.

Both players are partially informed, which is the main departure from CyGym. The defender sees only noisy audit signals. The attacker cannot observe resets, and because a reset benign node is behaviourally indistinguishable from one never compromised, must probe to estimate the current state. Compromise also propagates: a malicious agent can embed payloads in its outputs, infecting downstream agents along $\mathcal{E}$.

ANSG is the harder of the two deployment games for a budgeted build, for reasons the theory anticipates. Payoffs require full rollouts of a partially observable game, so simulation dominates iteration cost and the premise of Section 6 holds strongly. Equilibrium supports are not known in advance, so B^* must be estimated from bootstrap replicates rather than read off a solved game. And no ground-truth equilibrium is computable, so the certificate of Corollary 1 is the only available quality measure rather than a diagnostic checked against a known answer.

A.5 Budget sweep on Cyberwheel

When the pool is flat, every arm that buys cells returns essentially the same equilibrium, and budgeted acquisition has nothing to separate. Figure 7 repeats the budget sweep of Section 6.2 on Cyberwheel [26]. Uniform sampling, IGS, MRFS, and both RWPS variants all sit within about 0.015 of zero exploitability at every budget, with overlapping bands from budget 2 onward. The one visible separation is at budget 1, where RWPS (uniform) is highest of the buying arms. GS/PSP/PS-REG-M is constant near 0.16 because, as on CyGym and ANSG, its sampling schedule cannot afford a single cell and it returns the history-only estimate. The pool is flat in a measurable sense: on this 6×6 pool the payoff signal-to-noise ratio is below 1 for both roles, so no cell’s simulated value is distinguishable from rollout noise.

This is the outcome Corollary 2 predicts for a matrix on which no cell is much more deviation-relevant than another: if the surrogate is already accurate everywhere, which cells are bought barely changes the equilibrium. We report it as the regime in which RWPS offers no advantage over simple baselines, and in which it also costs nothing.

A.6 Implementation and computational procedure

A build with budget B runs R rounds. Each round refits the ensemble on the whole cache, predicts $\hat{u}_{ij}$ and $\hat{\sigma}_{ij}$ on every unsimulated cell, re-solves B_{boot} bootstrap realisations of the game to obtain the smoothed marginals, and then simulates about B/R cells, each uniform with probability ε_x and otherwise the highest-scoring. The surrogate is therefore refit $R + 1$ times per build, once per round and once more before the matrices are committed. Only the simulation step consumes budget; the bootstrap costs solves alone.

The cache persists across PSRO iterations, so a fit at iteration k trains on every cell bought earlier, and cached values always override the surrogate. Repeat visits, which arise only from exploration draws, are averaged into a running mean. The opening fit of each build warm-starts from the previous build’s weights, which leaves final exploitability unchanged ($p = 0.81$ informative,

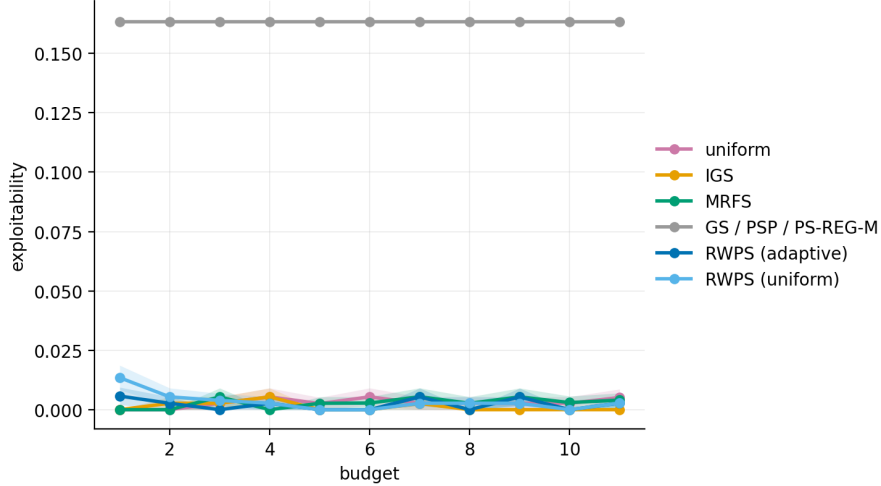


Figure 7: Cyberwheel budget sweep. 6×6 pool with a 5×5 history pre-cached for every arm, leaving 11 cells to buy; 16 seeds. Exploitability is computed against the stored payoff matrix of the pool. Bands are ± 1 standard error. All buying arms coincide within their bands; GS/PSP/PS-REG-M buys nothing and returns the history-only estimate.

$p = 0.62$ Blotto, eight seeds) and cuts build time by 17–20%; the later rounds of a build are fit cold, because the cache roughly doubles between rounds and a short warm schedule does not track that. Until the cache holds eight labels, a rank-one row-and-column model stands in for the ensemble, so the first round on a fresh pool is close to uninformed.

Each ensemble member is an MLP mapping the concatenated embeddings of a cell’s two strategies to both players’ payoffs. Member m trains on its own bootstrap resample of the cache (n cells drawn with replacement) with its own seed, which is the only source of disagreement between members; the loss is mean squared error on standardised targets, optimised full-batch with Adam for a fixed number of epochs, with no validation split or early stopping. This ensemble bootstrap measures uncertainty about individual payoffs. The separate acquisition bootstrap resamples 80% of each pool’s strategies and re-solves the game, and so measures uncertainty about the equilibrium.

On the synthetic games the restricted game is solved by fictitious play, whose residual regret is carried in every bound as ϵ_{solve} ; the CyGym, ANSG and Cyberwheel sweeps use CyGym’s solver, support enumeration with a Lemke–Howson fallback. Those sweeps also embed strategies by identity tags, one-hot vectors per strategy, since building a behavioural probe bank for each simulator was out of scope; with tags the surrogate learns only row and column effects. On ANSG, 36 cached cells were enough to trigger the MLP, which then filled the 13 unseen cells worse than a flat mean, so that sweep keeps every RWPS arm on the rank-one model throughout. Table 6 lists the settings.

A.7 Embedding Analysis

The latent-quality game admits a controlled ablation of the fingerprint. Holding the payoff matrices fixed and redrawing the fingerprints independently of latent quality yields a game identical in every respect except that the embedding carries no payoff information. Table 7 repeats the growing-pool comparison of Section 6.1 on both versions.

Every method that does not consult the embedding is unchanged to the reported precision, which is a check on the construction as much as a result. The estimator is the only row that moves, so 0.015 is a direct measurement of what payoff-informative fingerprints are worth in this game.

Table 6: Default settings unless stated otherwise; epochs are cold / warm fits.

Setting	Value
Ensemble	$M = 3$ MLPs, 2×48 hidden
Training	Adam, lr 5×10^{-3} ; 60 / 10 epochs
Rounds and batch	$R = 3$, $c = \lceil B/3 \rceil$
Bootstrap	$B_{\text{boot}} = 32$, 80% of each pool
Smoothing, exploration	$\nu = 0.25$, $\varepsilon_x = 0.15$
Rank-one fallback	below 8 cached labels
Fictitious play	400 final, 200 in bootstrap
Rollouts	$N_{\text{MC}} = 4$, $\sigma = 0.10$
Synthetic games	21×21 , payoffs in $[0, 1]$
Probe bank	$P = 32$ states per role

Table 7: Growing-pool PSRO on games with identical payoffs and different fingerprints; final exploitability, mean over five seeds, all budgeted methods at a budget of 256. Only the estimator reads fingerprints, so only its row moves.

Payoff build	informative	uninformative	Δ
RWPS (ours)	0.045	0.060	+0.015
MRFS	0.053	0.053	0
IGS	0.082	0.082	0
GS / PSP / PS-REG-M	0.269	0.269	0
<i>full, unbudgeted</i>	0.035	0.035	0

The estimator’s advantage over uniform sampling splits into a part that survives an uninformative embedding, $0.092 \rightarrow 0.060$, and a part attributable to generalisation across embedding space, $0.060 \rightarrow 0.045$. The first is the larger. With fingerprints that are pure noise the surrogate still learns row and column effects from cells already simulated: it learns that a given strategy tends to score well or badly against the pool, which requires no embedding at all, and that alone recovers most of the gap. This is why the rank-one row-and-column fallback of Appendix A.6 is not merely a cold-start device: it is a substantial fraction of what the surrogate does throughout.

It also explains where the estimator stops winning. On the uninformative game MRFS reaches 0.053 against the estimator’s 0.060, inside one standard deviation. MRFS obtains the same row-and-column structure directly, by expanding a profile into its unilateral deviations, and needs no embedding to do it. When the embedding contributes nothing there is nothing left for the estimator to win with, and the two methods are measuring the same thing by different means. We report this as the intended outcome of the ablation rather than as a defeat: the uninformative game exists to isolate the embedding term, and a near-tie with a model-free method is the correct result when that term is zero by construction.

A.8 Coverage iteration and asymptotic guarantee

Corollary 2 identifies the deviation-relevant set B^* and Corollary 3 prices a target certificate against it. This appendix turns the pair into an iteration and gives the asymptotic guarantee that follows. The finite-budget result the paper relies on is Theorem 1, which does not depend on anything here.

Corollary 1 reports what a completed build certifies; run backwards, it answers how much simulation buys a stated guarantee, before any of it is spent.

We do not treat B^* as a prediction that has to be got right. Instead we treat it as the first iterate

of a procedure that comes with a free stopping test. The test is $w_{\text{sur}} = 0$, which holds exactly when the deviation-relevant set of the returned $(\hat{p}, \hat{q})$ has been covered, and which is a cache-membership query costing no rollouts. This dissolves the apparent circularity in reading B^* as an a-priori budget. The supports it needs are a property of the solve that it funds, but the procedure does not need them to be correct, only to be verified at the end.

Proposition 1 (Coverage iteration). *From any cache, iterate: solve $(\hat{D}_t, \hat{A}_t)$ for $(\hat{p}_t, \hat{q}_t)$; simulate every uncached cell of $\{1:n_D\} \times \text{supp}(\hat{q}_t) \cup \text{supp}(\hat{p}_t) \times \{1:n_A\}$; refit and repeat, halting when $w_{\text{sur}}(\hat{p}_t) = w_{\text{sur}}(\hat{q}_t) = 0$. Then*

- (i) *the procedure halts after at most $n_D + n_A$ rounds;*
- (ii) *at halting the hypothesis of Corollary 2 holds for the equilibrium actually returned, so $\varepsilon \leq \varepsilon_{\text{solve}} + 2\zeta(m, \delta)$ with no dependence on β ;*
- (iii) *total simulation never exceeds $n_D n_A$ cells, so the procedure is never more expensive than the full rebuild it replaces.*

Halting is by exhaustion of rows and columns, so the sequence of mixtures need not converge, and an oscillating equilibrium costs additional rounds and nothing else. The procedure is sound and terminating but not minimal, since an intermediate mixture can pull in rows for a support that later vacates. Note that the two error directions are not symmetric. Overshooting the support buys unnecessary cells while keeping the guarantee, whereas undershooting forfeits it, so we seed the first iterate with the *union* of the bootstrap supports rather than with a threshold on their average.

Theorem 3 (Convergence under coverage). *Consider a sequence of builds in which the iteration of Proposition 1 is run to halting whenever the returned mixture changes, and in which every cell of the coverage set continues to be sampled. Then, on the event of probability at least $1 - \delta$ on which Theorem 1 holds,*

- (i) *at most $n_D + n_A$ coverage extensions occur over the entire sequence, after which $w_{\text{sur}}(\hat{p}) = w_{\text{sur}}(\hat{q}) = 0$ holds permanently;*
- (ii) *thereafter $\varepsilon \leq \varepsilon_{\text{solve}} + 2\zeta(m, \delta)$, and $\varepsilon \rightarrow \varepsilon_{\text{solve}}$ as $m \rightarrow \infty$;*
- (iii) *hence, with the restricted game solved exactly, the returned mixture converges to an exact equilibrium of that game, while the $n_D n_A - B^*$ surrogate-filled cells are never corrected and may remain arbitrarily wrong.*

A single confidence event supports every m here, which is what the time-uniform radius buys us: a fixed-design bound holds at one prescribed sample size, so letting $m \rightarrow \infty$ under it would require a union over sample sizes. Neither half of the result is an assumption. We reach coverage in finitely many rounds using a stopping test that is a cache query, and the vanishing of ζ follows from continued sampling of the cells that coverage identifies. Coverage decides which cells convergence is possible on and visits decide how fast, so spending alone does not substitute for coverage. The contrast with Section A.3.3 is one of cost: that argument needs forced exploration to visit every cell infinitely often, $\Theta(n_D n_A)$ cells, whereas this one needs only the $\Theta(n_D s_A + n_A s_D)$ cells of the coverage set.

Coverage guarantees the bound, not the solve. Driving w_{sur} to zero removes β from Theorem 1 by construction, so the bound holds whatever the surrogate predicts. The certificate no longer depends on surrogate quality. The surrogate has not ceased to matter. Which equilibrium the solver arrives at is still a function of the fill, because the solver reads the complete matrix and the unsimulated cells are exactly the cells that determined they were unplayed. The surrogate therefore controls

rounds to termination rather than the validity of what the iteration returns. The $\beta \cdot w_{\text{sur}}$ term cannot express this role, since coverage sets it to zero. At B^* coverage a surrogate fill and a flat fill carry the same guarantee, and Section 6 separates them empirically.

Coverage separates the budget into two parts that are priced independently. Coverage decides which cells are bought and removes $\beta \cdot w_{\text{sur}}$. Revisits decide how tight the bound is and shrink $2\zeta(m, \delta)$. Only the first is a question about the matrix, and it is the cheap one, costing $s_D + s_A$ cells per iteration under stable supports. The second is a visit count, and at small m it dominates, so a covered build certifies very little until m grows. A deployment can invert this: fix the exploitability bound required, take m to be the smallest visit count with $2\zeta(m, \delta)$ below it, and the rollout budget follows as the coverage set times m . Section 6 prices this.

The iteration trades a spending cap for a guarantee. A budget fixed as a fraction of the matrix bounds what a build can cost. The iteration instead spends what the game demands, which is $s_D + s_A$ cells per iteration under stable supports and, in the worst case, the full matrix. Under a hard rollout ceiling it may be truncated before it halts. Nothing is lost when that happens. Stopping with $w_{\text{sur}} > 0$ returns us to Theorem 1 with the representational term retained at the measured w_{sur} , which is the bound the estimator carries in any case. Halting is what removes β ; truncation degrades to the general bound rather than forfeiting it.

Fingerprint collisions do not defeat coverage. Theorem 2 constrains acquisition rules that read fingerprints, and its proof plants the trap in a column the rule never touches. Coverage admits no such column: $w_{\text{sur}}(\hat{p}) = 0$ requires $\text{supp}(\hat{p}) \times \{1:n_A\}$, so every attacker strategy is simulated against the defender’s support whatever its fingerprint reports, and the construction has nowhere to place the trap. If a trap’s true payoffs make it a profitable deviation against $\hat{p}$, covering reveals them, the re-solve moves mass onto it, w_{sur} becomes positive, and the next round completes its column. If it is profitable only against strategies outside $\text{supp}(\hat{p})$, then it does not enter $\max_j (A\hat{p})_j$ and is not a deviation by which the returned mixture can be exploited. Forced exploration remains load-bearing for the acquisition rule of Section 3.2, whose score is a function of the embedding, but it is not needed for the coverage guarantee.

Coverage costs $s_D + s_A$ cells per PSRO iteration whenever a new strategy pair leaves the supports unchanged. With a persistent cache the right baseline is not a $\Theta(n_D n_A)$ rebuild. It is the $2k+1$ cells that a pool of size k per side newly exposes, an incremental-caching accounting standard in EGTA [45, 47]. The result is immediate: those are the only cells of the deviation-relevant set the cache does not already hold, which is to say the only cells at which $w_{\text{sur}} \neq 0$. The condition is support stability rather than support size, and it is the regime PSRO enters near termination. The saving accrues late in a run rather than eroding.

Each result above attaches to a different part of the estimator, and none of them derives the acquisition rule. The surrogate fill steers *which* equilibrium is solved for. Coverage makes the *bound* on that equilibrium independent of the fill (Proposition 1). Corollary 1 *certifies* the result post hoc in settings where no ground truth exists, and Corollary 3 inverts that certificate into a budget.

One caveat is worth stating. Our acquisition rule targets value sensitivity, through W on support $\times$ support, rather than the deviation-relevant set, so it under-covers relative to Corollary 2, with the exploration draws supplying the remainder. Closing that gap with a rule that targets deviation coverage directly is an immediate direction for future work.

Table 8: Exploitability against batch size c and exploration probability ε_x , 20% budget, mean over four seeds. Batch size is the operative knob; ε_x is close to flat.

c	ε_x			
	0.00	0.05	0.15	0.30
<i>Latent, informative</i>				
1	0.018	0.029	0.019	0.028
4	0.030	0.054	0.056	0.027
16	0.014	0.031	0.037	0.042
64	0.041	0.028	0.033	0.074
<i>Blotto</i>				
1	0.073	0.042	0.078	0.074
4	0.086	0.074	0.107	0.106
16	0.164	0.164	0.194	0.160
64	0.163	0.139	0.214	0.167

A.9 Ablations and bound diagnostics

A.9.1 Error decomposition

Counterfactual decomposition (Figure 8) replaces one error source at a time with ground truth and re-solves. Error is representational at low budget and statistical at high, with the crossover below 20% on the informative game, between 20% and 40% on the arbitrary game, and near 40% on Blotto. This yields a tuning rule: below the crossover spend on coverage, above it on revisits.

The crossover need not be measured. Coverage and revisits buy different terms of Theorem 1 and are priced separately: coverage decides *which* cells and drives $\beta \cdot w_{\text{sur}}$ to zero, while revisits decide *how tight* and shrink $2\zeta(m, \delta)$. The rule in closed form is therefore to cover until $w_{\text{sur}} = 0$, which Proposition 1 bounds at $n_D + n_A$ rounds and costs $s_D + s_A$ cells per PSRO iteration under stable supports, and then to spend every remaining rollout revisiting covered cells until $2\zeta(m, \delta)$ meets the exploitability target. The two phases do not compete for the same cells: revisits fall only on cells coverage has already bought.

A.9.2 Batch size and exploration probability

The acquisition loop has two knobs. The batch size c is the number of cells bought between refits: $c = B$ commits the whole budget on a single ranking, $c = 1$ re-ranks after every cell. The exploration probability ε_x is the chance that a given selection ignores the score. Table 8 sweeps both at a 20% budget over four seeds.

Batch size is the parameter that matters, and it matters where the theory says it should. On Blotto, $c = 1$ reaches 0.073 against 0.163 at $c = 64$, a factor of 2.2; on the informative game the effect is within seed noise. The asymmetry follows from what a batch commits to: every cell in a batch is chosen from one ranking, so a large c spends that share of the budget on the surrogate’s state at the start of the batch. On a game the surrogate fits well that state is already good and the loss is small; on a combinatorial surface it is not, and re-ranking after each cell recovers most of the gap.

The exploration probability is close to flat, and $\varepsilon_x = 0$ is best or tied in several cells. This is the expected reading rather than a contradiction. Theorem 2 establishes that exploration is *necessary*, in the sense that no rule reading only fingerprints and the cache carries the asymptotic guarantee without it; it does not claim that exploration improves finite-budget exploitability on

a benign instance. These three games are benign in exactly that sense—no fingerprint collision is constructed—so exploration buys the guarantee and not the number, and its cost is the small budget share it diverts. The instance where it earns its keep is the twin-trap construction, reported separately in Section 6. We therefore keep $\varepsilon_x = 0.15$ as the default: it is insurance whose premium is measured here and whose payout is measured there.

A.9.3 Variants of the acquisition score

Table 3 compares our rule against prior work and against no score at all. This subsection varies our own rule instead, under the same protocol: identical surrogate, budget, batch schedule and forced-exploration rate, sixteen paired seeds, ε_{12} . None of these is a baseline drawn from the literature; each removes or replaces one ingredient of the score in Algorithm 1.

Table 9: Variants of our acquisition score, extending Table 3 with three further rules under the identical protocol; the first row repeats Table 3 for reference. Mean $\pm$ one standard error over sixteen paired seeds.

Acquisition score	Informative	Blotto
ν -smoothed product (ours)	0.024 $\pm$ 0.004	0.082 $\pm$ 0.011
bootstrap inclusion	0.035 $\pm$ 0.006	0.100 $\pm$ 0.017
training exposure	0.039 $\pm$ 0.005	0.105 $\pm$ 0.011
value-sensitivity ($\nu = 0$)	0.048 $\pm$ 0.010	0.096 $\pm$ 0.015

Value-sensitivity ($\nu = 0$) isolates the smoothing, and the ordering reverses across games. It is the rule used in earlier versions of this work: no smoothing, and the product returned inside the bootstrap average. Smoothing is worth 0.024 on the latent games ($p = 0.007$, 12/16 seeds) and 0.014 on Blotto, where it is not significant ($p = 0.348$, 9/16). It therefore does its work where the support estimate is sharp and the unsmoothed score concentrates on a handful of cells, not where the support is already broad—on Blotto, at supports (12, 12), the unsmoothed score already spreads over most of the matrix and there is little left for ν to do.

Bootstrap inclusion replaces equilibrium mass with inclusion frequency, the fraction of bootstrap replicates in which a strategy appears in the support at all, and sums the two roles rather than multiplying them. This is the discrete analogue of the probability-of-equilibrium criterion of Picheny et al. [28], with the bootstrap standing in for their Gaussian-process posterior. The sum is our departure from it: a product is supported only on $\text{supp}(\hat{p}) \times \text{supp}(\hat{q})$, whereas a sum scores every row against any plausible column, which is the shape of the deviation-relevant set of Corollary 2. It is the same neighbourhood idea as ν , learned from the bootstrap rather than imposed by a fixed constant, and it finishes second here—close enough that the choice between them is not settled by these games.

Training exposure is the PSRO-specific rule. Every strategy in the pool arrived as a best response to a known opponent mixture, so the cells pairing it with the opponents it was optimised against are cells whose values that optimisation has already partly determined; the cells it was never trained against are both less predictable and, by Corollary 2, deviation-relevant as soon as that strategy enters a support. The rule down-weights a cell by the training exposure $e(i, j)$, the weight the mixture that produced a_j placed on defender i , and symmetrically. It is the only rule considered anywhere in this paper that uses information no payoff-build baseline can see, since only the outer loop knows what each strategy was trained against, and it finishes fourth. We report it because the negative result is informative: a best response supplies one scalar constraint on a $\hat{p}$ -weighted average of a column, not per-cell information, and weighting cells by it asks the constraint to carry more

than it does. The constraint is better used inside the surrogate fit, which we have not attempted here.

A.10 Bound and certificate diagnostics

These three studies use fixed pools as a diagnostic bed rather than the growing-pool setting of Section 6. The first shows that the deviation-relevant set predicts cost before any budget is spent. The second breaks Figure 5 down by game. The third gives what the computable certificate costs in optimism. None is needed to read Table 1, Table 3, or Figure 5.

A.10.1 Predicting difficulty before spending

The results above are retrospective: they report what a budget bought. The practical question is the prospective one, whether a game’s cost can be known before committing to it. Corollary 2 answers it with $B^* = n_{DS_A} + n_{AS_D} - s_{DS_A}$, computable from bootstrap support estimates alone, and Table 10 evaluates it directly: simulate exactly the deviation-relevant set, surrogate elsewhere.

Two claims are tested. First, B^* *orders the games correctly*. The two latent games have supports of size 2 on each side and cost 80 cells, 18% of the matrix; Blotto’s supports come out at (12, 12) under this solver, giving $B^* = 360$ cells, 82%. That ordering is available before any budget is spent, and it matches the ordering the growing-pool results of Section 6.1 reveal. Equilibrium support size, not matrix size, is the operative quantity. Second, simulating exactly that set *delivers full-rebuild quality*: on the informative game coverage reaches 0.017 ± 0.003 against 0.033 ± 0.014 for a full rebuild, at 18% of the simulation, and on the arbitrary game it is statistically indistinguishable (0.038 ± 0.020). Coverage is therefore not merely equal to a full rebuild on the informative game, it is slightly ahead of it, and the reason is worth stating because it looks at first like an error. A full rebuild measures every one of the 441 cells independently at $N_{MC} = 4$ and keeps the statistical error of all of them. Coverage measures 80 cells and predicts the rest from a model fitted to everything observed so far. When that model class contains the truth, its predictions pool information across the matrix and carry *lower* variance than an individual four-rollout average, so replacing measurements by predictions can reduce error rather than add it. The latent games are the extreme case, since their payoff matrices are exactly rank one by construction (Section 5) and the surrogate’s row-and-column fallback is correctly specified for them. Applying a rank-one projection to a full rebuild’s own noisy measurements reproduces the effect on its own, halving realised exploitability, which confirms that the gain comes from denoising rather than from anything the acquisition rule does.

The effect tracks how well the model class contains the truth, and it disappears when that fails. On the arbitrary game the payoffs are the same but the fingerprints carry no information, so only row and column effects can be pooled and coverage is indistinguishable from a rebuild. On Blotto it is worse. The payoff surface there has effective rank above four, the surrogate cannot fit it, and at 82% coverage there is little left to save in any case.

A.10.2 Bound tightness by game

Figure 9 reports the same 240 runs as per-game overestimation factors. The sup-norm bound (d) overestimates realised ε by mean factors of 22.7, 20.0, and 11.7 on the three games; the support-weighted form (c) reduces these to 11.1, 7.2, and 4.3; the signed form (a) with the measured solver term to 5.8, 3.4, and 2.0. Mean ratios carry heavy right tails, since a run with near-zero realised ε inflates the ratio however tight the bound is in absolute terms, so the seed spreads are large. The ordering (d) > (c) > (a) is uniform across games and budgets.

Table 10: Deviation-relevant coverage (Corollary 2), mean $\pm$ SD over sixteen seeds, on 21×21 games (441 cells).

Game	supp.	B^* (%)	ε
Latent, informative	(2, 2)	80 (18)	0.017 ± 0.003
Latent, arbitrary	(2, 2)	80 (18)	0.038 ± 0.020
Blotto	(12, 12)	360 (82)	0.062 ± 0.034
<i>full rebuild, latent</i>		441 (100)	0.033 ± 0.014
<i>full rebuild, Blotto</i>		441 (100)	0.043 ± 0.021

A.10.3 The computable certificate, and the cost of optimism

Corollary 1 is evaluated in both of the roles Section 4 distinguishes. Used post hoc at B^* coverage, the computable certificate held on all 48 runs, with values of 0.935 ± 0.059 and 0.938 ± 0.075 on the latent games and 0.516 ± 0.032 on Blotto against realised ε of 0.017–0.062 (Table 10). At $m = 4$ visits its slack is dominated by the time-uniform concentration radius $2\zeta_{4,0.05} = 0.481$ and by mismatch between the estimated and true supports, and it contracts as $1/\sqrt{m}$ with revisits. Proposition 1 removes the second of those two sources by construction: iterating coverage to $w_{\text{sur}} = 0$ leaves the statistical radius as the only slack.

That remaining slack is the operative constraint, and it is a visit count rather than a coverage question. At $m = 4$ the bound a fully covered build can certify is $\varepsilon_{\text{solve}} + 0.481$, roughly half the payoff range, which is why the certificates above sit near 0.94 against realised ε of 0.017–0.062: the estimator was buying the right cells and not enough visits. Inverting $2\zeta(m, \delta)$ prices the alternative. A certificate of 0.25 needs $m = 16$ and, at the 80 cells of B^* coverage, 1,280 rollouts—below the 1,764 a full rebuild spends at $m = 4$. A certificate of 0.10 needs $m = 101$ and 8,080 rollouts, and one of 0.05 needs $m = 408$ and 32,640, some $18\times$ a full rebuild. Coverage is cheap and the certificate is not; a deployment should choose the bound it needs and let the visit count follow, rather than choosing a cell budget and discovering what it certifies. Used inside the solve, R-max style, optimism certified only after simulating 90–97% of the matrix (395–430 of 441 cells), because every unsimulated cell looks maximal and the support chases fresh cells. The division of labour is therefore a three-way one: the surrogate fill decides which equilibrium is solved for, coverage of the deviation-relevant set makes the bound on that equilibrium independent of the fill, and optimism only certifies it. The B^* runs above isolate the middle role—every one of them carries the same β -free guarantee—while the exploitability they reach still separates surrogate fill from the flat fill of the uniform baseline, which is the first role.

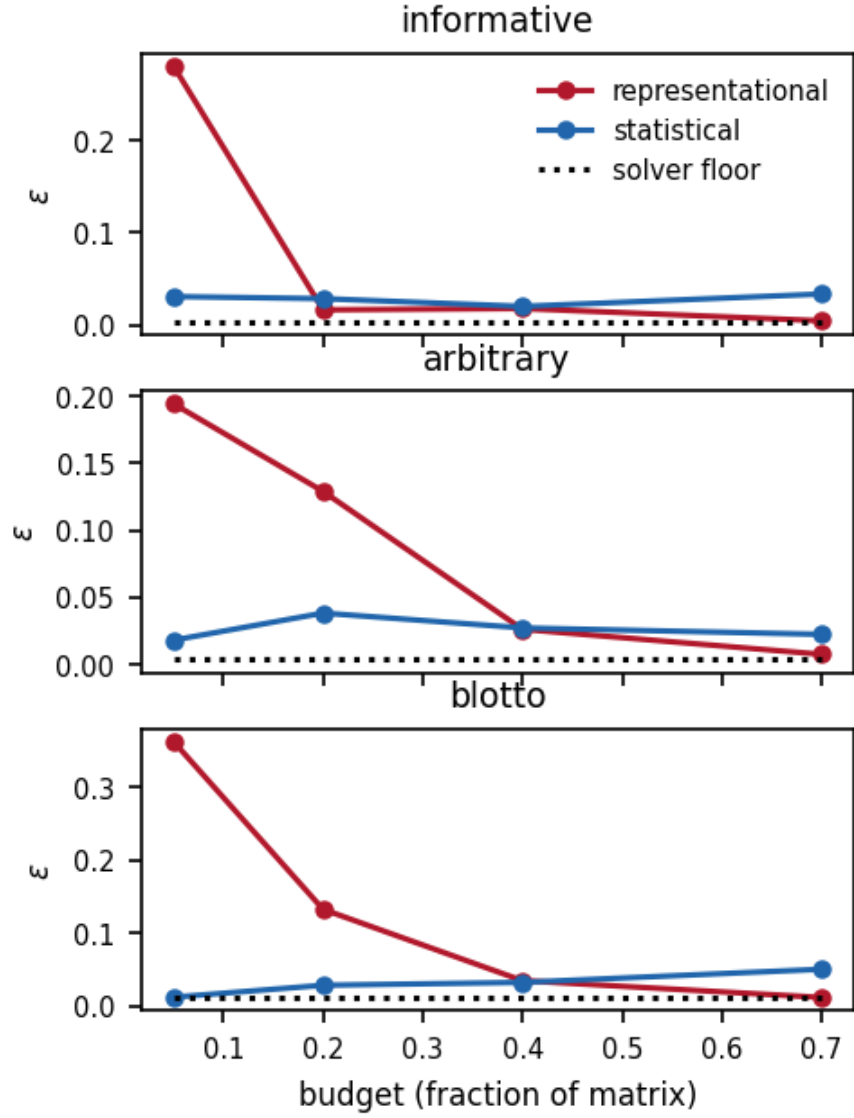


Figure 8: Counterfactual decomposition of ϵ into statistical and representational parts (seed means; dotted line is the solver floor).

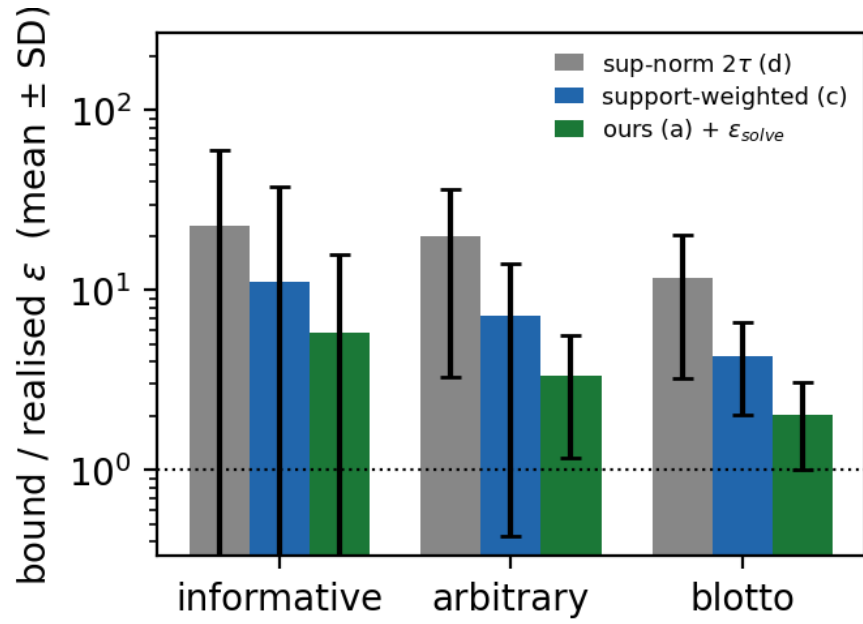


Figure 9: Overestimation factor, bound over realised ε , per game (mean $\pm$ SD over the 240 budget-sweep runs, log scale). Labels refer to the forms of Eq. (3); our bound is 4–6 $\times$ tighter than the sup-norm bound.